175 PTAS

MI COMPUTER

52

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR



Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen V - Fascículo 52

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones Francisco Martín
Asesor técnico:	Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-007-4 (tomo 5)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 098501
Impreso en España - Printed in Spain - Enero 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



Especialidad japonesa

Los ordenadores de bolsillo se utilizan para diversas aplicaciones serias. Esta vez comparamos varias máquinas de la gama Casio

Las calculadoras programables se pueden utilizar en muchas aplicaciones similares, pero los ordenadores de bolsillo tienen la ventaja definitiva de que pueden emplear BASIC y manipular texto, mientras que las calculadoras programables utilizan sus propios lenguajes especiales (que con frecuencia se asemejan al lenguaje máquina) y son aptas sólo para tareas numéricas. Los ordenadores de bolsillo son ideales para aquellas actividades en las que es necesario efectuar cálculos que implican una fórmula establecida. Tales cálculos suelen ser repetitivos y tediosos; los usuarios de ordenadores de bolsillo pueden escribir sus propios programas para tratar este tipo de cálculos.

De la gama de ordenadores de bolsillo Casio, bien establecida, el más barato es el FX-720P. Sus dimensiones son de 165×85×15 mm y pesa sólo 210 g. El FX-720P posee un teclado cuyas dimensiones representan algo más de la mitad de las de un teclado normal. Utiliza el trazado QWERTY convencional, con la excepción de que todas las filas están alineadas en vez de seguir la disposición alternada habitual.

Cada tecla alfabética puede producir otros dos caracteres (como signos de puntuación) y palabras clave de BASIC, cuando se utilizan junto con las teclas Shift o Function. Sin embargo, a diferencia de otros teclados, estas teclas no se emplean simultáneamente con la tecla alfabética (ALPHA); se pulsa primero una y se suelta antes de pulsar la tecla apropiada. Ello hace que resulte fácil trabajar en el teclado con una sola mano (dejando la otra libre para sostener el ordenador).

El FX-720P utiliza una visualización en cristal líquido (LCD) con 12 caracteres en una única línea, y las líneas más largas se visualizan pulsando dos teclas del cursor para desplazamiento a izquierda o derecha. Moviendo con el pulgar una ruedecilla que hay a uno de los lados de la visualización se ajusta el contraste de la LCD de modo que se adapte a los distintos ángulos de vista y niveles de iluminación. Si el ordenador no se utiliza durante seis minutos, la visualización se apaga para ahorrar energía.

El FX-720P se vende con una escasa memoria de dos Kbytes. No obstante, toda la memoria para el usuario se proporciona en forma de cartuchos enchufables que conservan su contenido cuando se quitan de la máquina. Un usuario puede, en consecuencia, almacenar distintos programas y datos en cartuchos separados, que se enchufarán cuando sea necesario.

Dado que los cartuchos pueden retener tan pocos datos, se llenan enseguida, debido especialmente a que el ordenador permite retener en la memoria hasta 10 programas en BASIC al mismo tiempo. Se debe seleccionar una modalidad especial para dar entrada a un programa en BASIC y se necesita otra para ejecutarlo. La versión de BASIC del FX-720P es sorprendentemente buena para tratarse de una máquina tan pequeña: incluye casi todas las instrucciones y funciones estándares, a excepción de CHR$ y ASC. Una instrucción simple, BEEP, permite producir dos sonidos diferentes.

Se considera que el FX-720P se halla en el extremo inferior de la gama Casio de ordenadores personales y, por consiguiente, la mayoría de sus funciones se ven superadas en los otros modelos. Existe, no obstante, una característica que es exclusiva de esta máquina. Se trata del *Data bank* (banco de datos), que es un pequeño programa de base de datos. Utilizando este programa se puede entrar fácilmente información del tipo nombres y direcciones, gastos o citas. El programa buscará cualquier ítem de información e incluso permite diferentes niveles de búsqueda; por ejemplo, si con una búsqueda se ha encontrado una gran cantidad de ítems, se puede realizar otra búsqueda en estos mismos utilizando otro criterio. Esta característica hace que la máquina sea una competidora directa del Psion Organiser, aunque el FX-720P cuesta menos y es más versátil.

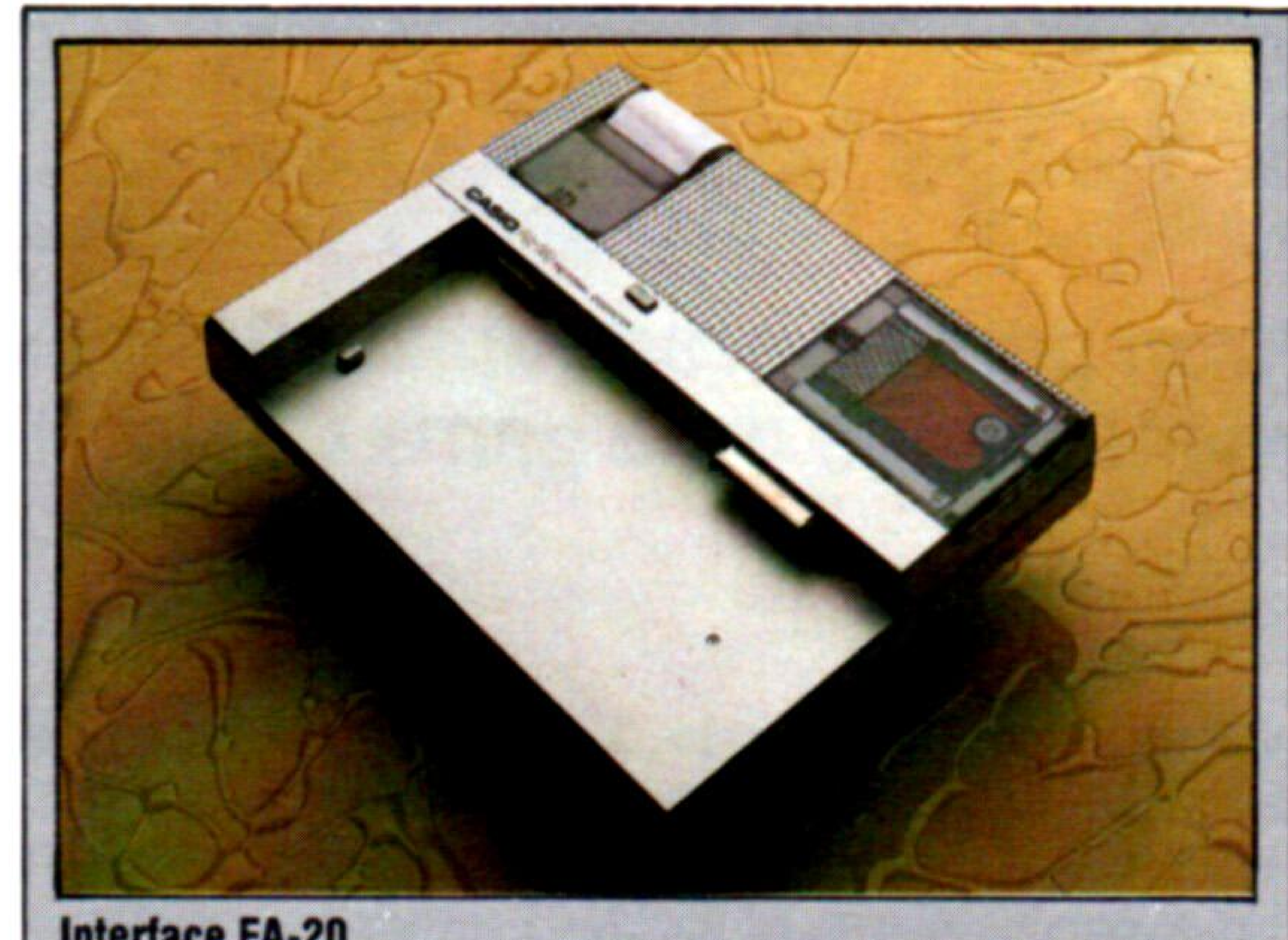
Casio FX-750P y PB-700

Si el FX-720P parece estar dirigido a los hombres de negocios, el siguiente modelo de la gama Casio, el FX-750P, pretende cubrir las necesidades de científicos e ingenieros. Ofrece una visualización de



Chris Stevens

Modelo económico
El ordenador personal FX-720P de Casio tiene 2 K de memoria en cartuchos enchufables, una visualización en cristal líquido de una línea y un juego completo de teclas alfabéticas y numéricas. Se pueden enchufar cartuchos adicionales para proporcionar memoria extra



Interface FA-20
Cada uno de los ordenadores de bolsillo de la gama Casio se puede acoplar a una interface de este tipo. Este modelo permite albergar limpiamente el FX-750P. La FA-20 viene con pilas recargables y un adaptador de corriente

Chris Stevens

24 caracteres y un teclado ligeramente mejorado. Viene con un cartucho de memoria de dos Kbytes, del mismo tipo que utiliza el FX-720P, pero además posee una ranura para un segundo cartucho. Esto significa que su capacidad de memoria se puede ampliar hasta ocho Kbytes. La máquina mide 185 mm de ancho y pesa apenas 250 g. A diferencia del FX-720P, las teclas Shift y Function se deben utilizar al mismo tiempo que la tecla alfabética.

En diez de las teclas hay programadas importantes constantes: la velocidad de la luz, la aceleración gravitatoria de la Tierra, la constante de Planck, la constante de Boltzmann, la constante de Avogadro, la carga elemental, la masa atómica, la masa electrónica, la constante gravitatoria y el volumen molar. Los físicos y los químicos utilizan con mucha frecuencia estos números en sus cálculos y el hecho de tenerlos fijos en la memoria les evita el problema de tener que recordarlos y digitarlos. El ordenador también incluye funciones de las que el BASIC normalmente no dispone: seis funciones hiperbólicas y un juego de funciones estadísticas que incluyen desviación estándar, regresión lineal y correlación. El BASIC está mejorado en otros varios sentidos de modo tal que no sería inapropiado para muchos ordenadores personales.

El tercer ordenador de bolsillo de Casio es el recientemente lanzado PB-700. Posee una visualización de 20 caracteres por cuatro líneas y cuatro Kbytes de memoria. La memoria se puede ampliar hasta 16 Kbytes utilizando paquetes de memoria que se enchufan en el lado de abajo del ordenador. Tiene casi 200 mm de ancho y pesa 315 g. Al igual que los otros ordenadores de bolsillo, viene con una carcasa blanda de protección.

El PB-700 tiene un teclado semejante al del FX-750P, con la excepción de que tiene una tecla Caps en lugar de la tecla Function. Mientras se mantiene pulsada esta tecla, el ordenador produce letras en minúscula. La versión de BASIC de la máquina es similar a la versión del FX-750P, si bien posee algunas instrucciones extras para dibujar gráficos en la LCD, que tiene una resolución de 32 por 160 puntos.

El BASIC incluye, asimismo, instrucciones para trazar gráficos utilizando la unidad interface opcional de impresora-plotter en color y cassette, denominada FA-10. Ésta se acopla al ordenador. Utiliza cuatro bolígrafos de distintos colores para dibujar texto y gráficos en un papel de 115 mm de ancho. Las instrucciones del BASIC permiten que la impresora-plotter dibuje líneas, círculos y ejes para gráficos.

La FA-10 también lleva incorporada una interface de cassette que le permite utilizar una grabadora de cassette normal para guardar programas y datos. Considerando lo pequeña que es la memoria del ordenador, ésta es una facilidad muy útil. Al instalar en la impresora-plotter una diminuta grabadora de cassette se obtiene un sistema informático completo lo suficientemente pequeño como para poderlo llevar a cualquier parte. También hay a la venta una interface Centronics separada que permite utilizar con el PB-700 las impresoras para ordenador estándares.

Para el FX-720P y el FX-750P también hay impresoras e interfaces de cassette. El FX-720P utiliza la impresora FP-12S. Una unidad separada, la FA-3, actúa como interface para una grabadora de cassette. El FX-750P utiliza la FA-20, una diminuta impresora e interface de cassette combinadas. Tiene un espacio para almacenar un cartucho de memoria y viene con pilas recargables, un transformador de corriente y una carcasa protectora.

Los tres ordenadores se suministran con manuales exhaustivos. Éstos contienen todo lo necesario; pero es obvio que están traducidos del japonés y en muchos puntos la redacción resulta un tanto extraña. No es que sean inutilizables, sino que pueden resultar confusos para el principiante. Otro problema es que para estos ordenadores casi no hay software. Los usuarios tendrán que escribirse sus propios programas o bien copiar los programas de muestra que vienen listados en el manual de cada máquina.

PRODUCTO	DIMENSIONES	PESO	MEMORIA	VISUAL.	OBSERVACIONES
FX-720P	165×85×15 mm	210 g	2 K	1×12 car.	Base de datos incorporada. Teclado numérico
FX-750P	185×85×15 mm	250 g	2 K	1×24 car.	Ranura para segundo cartucho. Constantes físicas ya programadas. BASIC ampliado
PB-700	200×85×15 mm	315 g	4 K	4×20 car.	Capacidad para gráficos. Impresora-plotter



Tortuga juguetona

Vamos a analizar aquí una modalidad alternativa a la inmediata: la de edición, que permite crear nuevas instrucciones LOGO

Tal vez los *procedimientos* no les resulten familiares a muchos programadores de micros, pero el concepto de procedimiento está presente en "procedimientos de instrucciones" cotidianos tales como recetas de cocina y patrones de labores de punto. En este capítulo vamos a crear un procedimiento en LOGO al cual llamaremos CUADRADO. (A diferencia de las instrucciones de LOGO, que siempre se deben digitar en mayúsculas, los nombres de los procedimientos se pueden digitar en mayúsculas o bien en minúsculas.) Comenzamos por digitar:

```
EDIT CUADRADO
```

Se limpiará la pantalla y luego aparecerá el mensaje TO CUADRADO en la parte superior de la visualización a modo de recordatorio del nombre del procedimiento, y en la parte inferior aparecerá EDIT CTRL-C TO DEFINE CTRL-G TO ABORT (EDIT CTRL-C para definir CTRL-G para abortar). Este mensaje un tanto críptico sirve de ayuda para ir guiando sus acciones mientras esté en modalidad de "edición". EDIT (edición) simplemente le informa que el LOGO ya no está en modalidad inmediata sino que espera a que se cree un procedimiento. Una vez hecho esto y estando ya listo para almacenar su procedimiento, pulsando Control-C se definirá (registrará) el listado, mientras que Control-G permitirá abortar el procedimiento y volver a empezar.

En la modalidad de edición se puede digitar todo lo que se desee, pero en la modalidad no se obedecerá ninguna de las instrucciones. El LOGO simplemente toma nota de sus instrucciones y las almacena en su "diccionario" bajo el nombre que se le haya dado al procedimiento. Complete la definición del procedimiento CUADRADO entrando las siguientes instrucciones:

```
TO CUADRADO
  REPEAT 4 [FD 50 RT 90]
END
```

Ahora puede decirle al LOGO que defina este procedimiento (es decir, que lo recuerde) digitando Control-C; entonces recibirá el mensaje CUADRADO DEFINED (definido). Si ha cometido un error en su definición, puede abortar todo el proceso digitando Control-G, en cuyo caso debe volver a empezar, o bien puede utilizar el editor de pantalla completa de la modalidad de edición para efectuar correcciones. Éste permite usar las teclas del cursor para desplazarse hasta cualquier punto del procedimiento e insertar o eliminar caracteres. (El editor de línea de la modalidad inmediata permite efectuar correcciones sólo en la última línea entrada.) El LOGO posee muchas más instrucciones que hacen que la edición de procedimientos extensos resulte simple; nos ocuparemos de ellas más adelante.

Ahora que ha definido con todo éxito su procedimiento, veamos cómo funciona. Entre DRAW para acceder a la pantalla de gráficos, luego entre CUADRADO; ahora se obedecerán las instrucciones de la definición del procedimiento y la tortuga dibujará el cuadrado. Como puede ver, los procedimientos se utilizan exactamente de la misma manera que las instrucciones básicas del LOGO; sencillamente se entra el nombre del procedimiento y se ejecutan las instrucciones previamente definidas. Las instrucciones originales que están presentes cuando se carga el LOGO se denominan *primitivas*. Una vez que se le ha "enseñado" al lenguaje un nuevo procedimiento, éste se puede emplear exactamente de la misma forma que una primitiva. En otras palabras, el LOGO es un lenguaje "extensible", de modo que puede ser confeccionado a medida para que se adapte a las propias necesidades.

Si su procedimiento no hace lo que esperaba que hiciera, es muy fácil modificar la definición. Tal como lo hemos definido, CUADRADO dibuja un cuadrado con lados de 50 unidades de largo. Vamos a suponer que se prefiera un cuadrado más pequeño, supongamos que de 30 unidades de lado. Retornemos al editor digitando:

```
EDIT CUADRADO
```

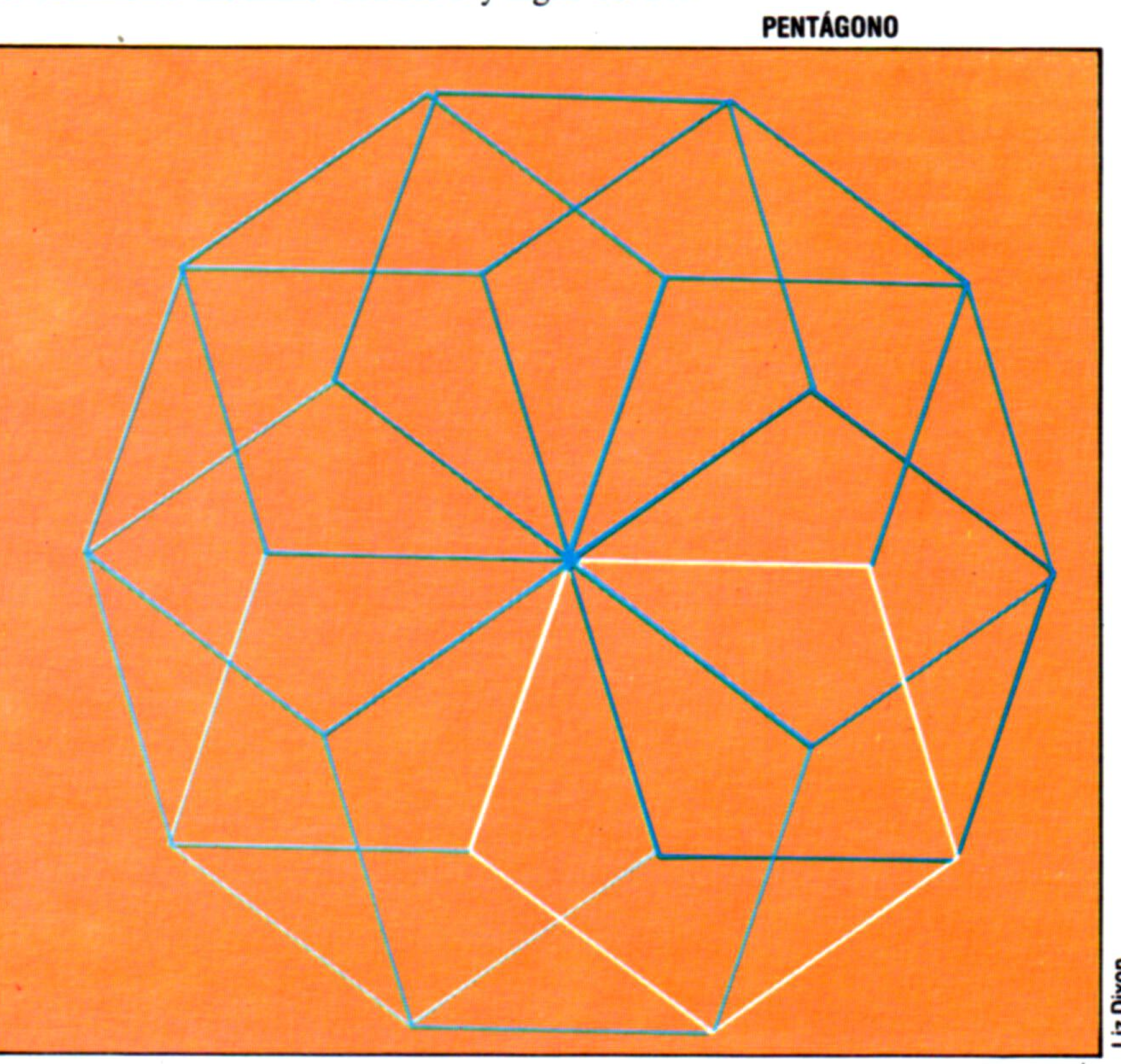
Ahora se visualiza el texto de que consta el procedimiento y se puede utilizar el editor de pantalla para cambiar 50 por 30. Después de hecho esto, redefina el procedimiento mediante Control-C y digite CUADRA-



Abreviaturas

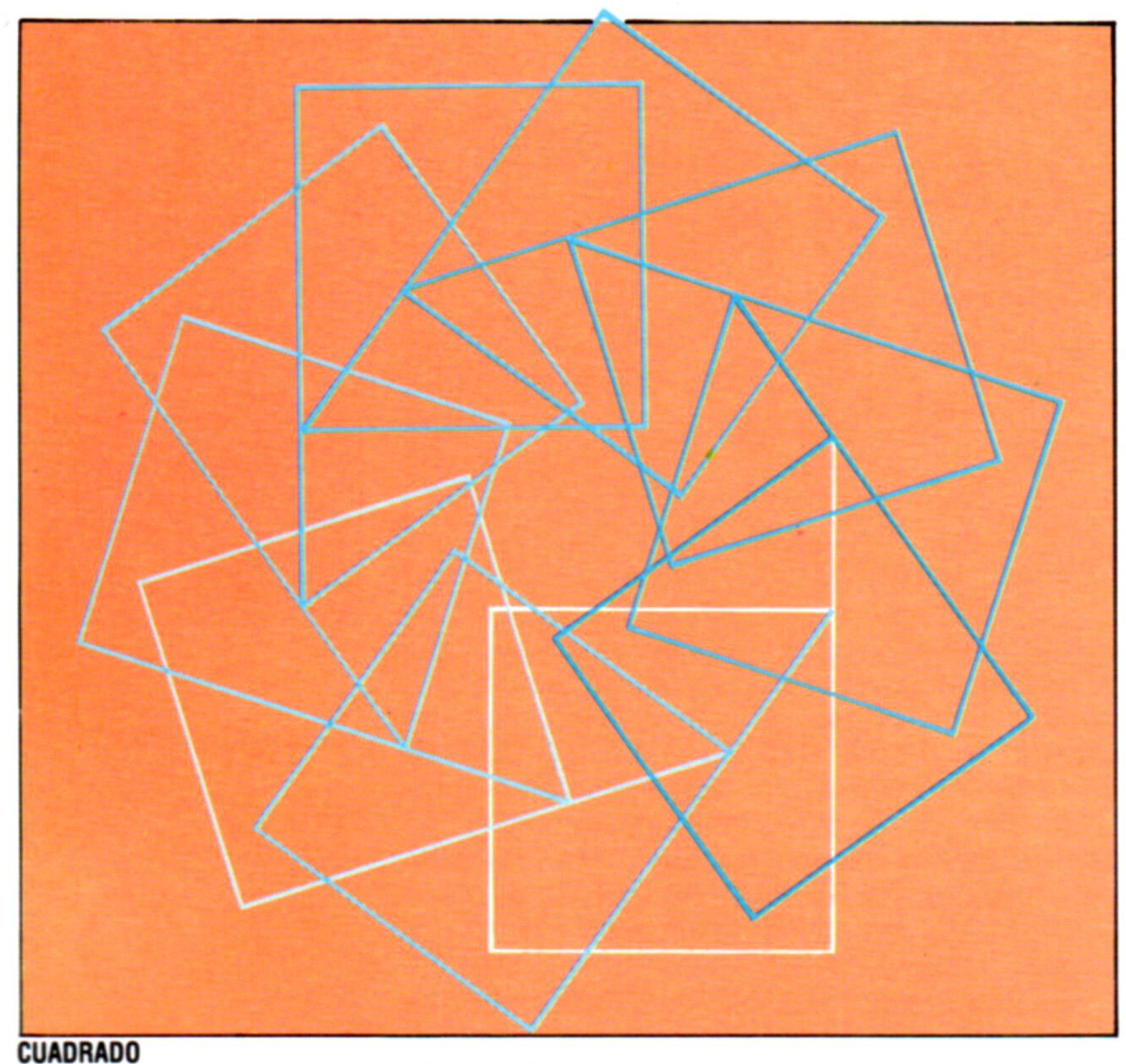
EDIT	ED
HIDETURTLE	HT
SHOWTURTLE	ST

PENTÁGONO



Liz Dixon





CUADRADO

En el diagrama de la página contigua le mostraremos los resultados que da la tortuga. Los ejemplos que ofrecimos en el capítulo anterior estaban diseñados para que se experimentara con ellos; quizá sus intentos por dibujar las diversas formas sugeridas le hayan llevado a algunos resultados inesperados. Para dibujar un triángulo, tal vez primero haya probado con algo más o menos así:

```
REPEAT 3 [FD 50 RT 60]
```

¡y entonces habrá descubierto que ha creado la mitad de un hexágono! Si intentó dibujar la estrella de cinco puntas, ciertamente le habrá resultado mucho más difícil de lo que pensaba. Cuando uno trata con problemas de este tipo, suele ser de ayuda "jugar a la tortuga", imaginando que *uno* es la tortuga que se va moviendo por ahí. En realidad, se dice que es posible distinguir entre un programador de BASIC y un programador de LOGO ¡mirando cómo mueven los hombros mientras programan! Imagínese que dibuja cualquier forma cerrada desde el punto de vista de la tortuga. La tortuga ha de dar la vuelta completa alrededor de la forma y debe terminar en el mismo lugar donde empezó, mirando hacia la dirección original. De modo que debe girar a través de un múltiplo de 360º. Si la forma es "convexa" (si ninguno de sus ángulos interiores es mayor de 180º), entonces la tortuga habrá girado a través de 360º exactamente. En el caso de un triángulo, son tres los giros a efectuar, de modo que cada uno de ellos ha de ser de 360/3=120º. A partir de esto, se puede deducir que la instrucción correcta para dibujar un triángulo es:

```
REPEAT 3 [FD 50 RT 120]
```

Incluso para polígonos no convexos, como las formas en estrella, se aplica el mismo principio; la única diferencia es que en este caso el ángulo total descrito es un número entero múltiplo de 360º (porque se está creando más de una forma completa).

Estos principios son generales (no se aplican simplemente a los polígonos) y componen lo que se conoce como el *teorema del recorrido total de la tortuga*. Si examina la estrella de seis puntas desde el punto de vista de la tortuga, verá que ésta no puede dibujar la forma si sólo se mueve hacia adelante y gira describiendo el mismo ángulo cada vez; se requiere un procedimiento más complicado.

DO para asegurarse de que ahora es del tamaño que se deseaba. A modo de ejercicio, intente escribir procedimientos para dibujar las formas que creó en el capítulo anterior (véase p. 1012): triángulos, rectángulos, pentágonos, estrellas, etc.

Ahora trate de emplear sus procedimientos con la instrucción REPEAT. Por ejemplo, defina PENT como:

```
REPEAT 5 [FD 50 RT 72]
```

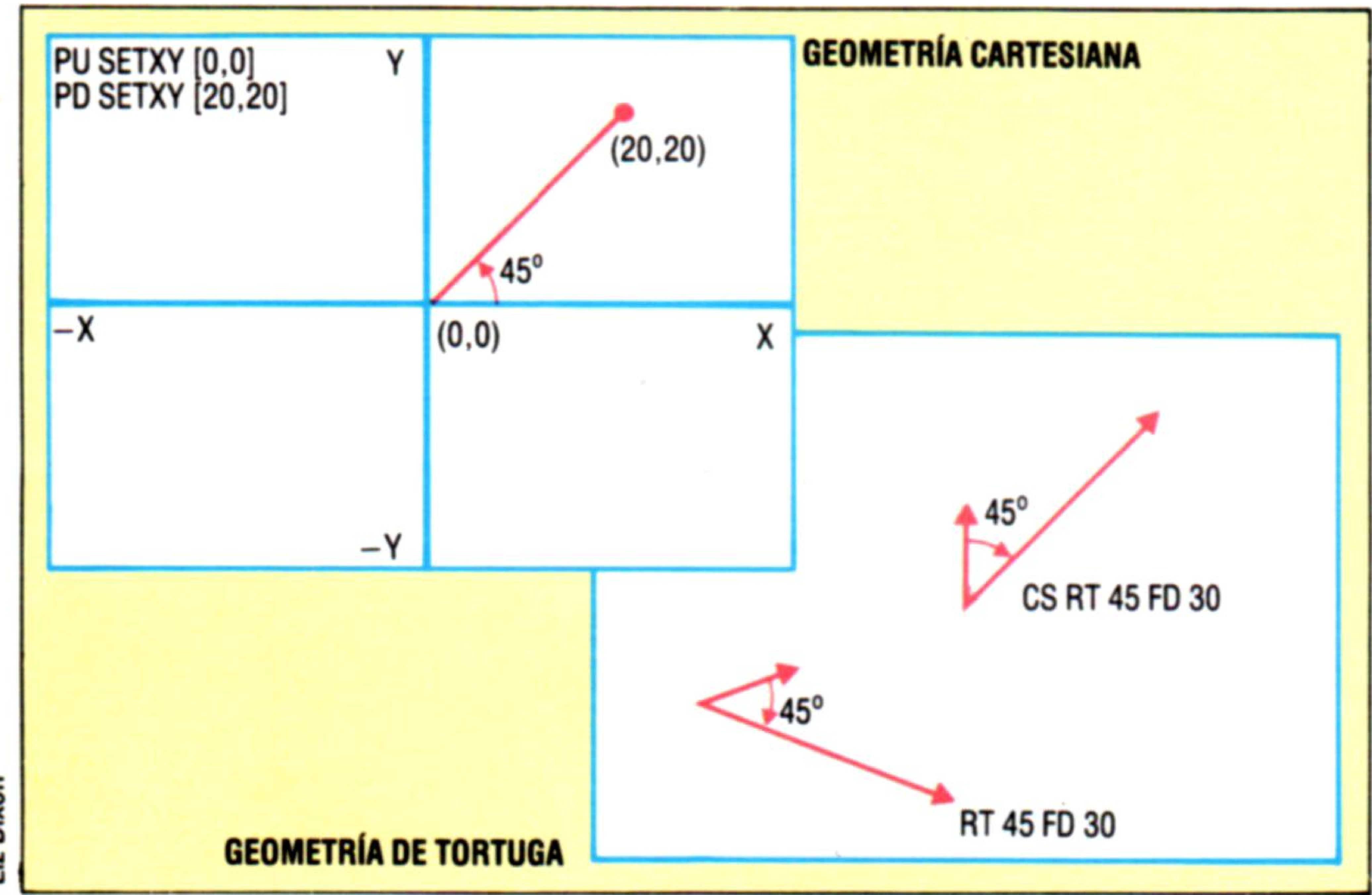
y después pruebe:

```
REPEAT 10 [PENT RT 36]
```

Puede experimentar con otras formas de esta misma manera, y pronto descubrirá que mediante la utilización de REPEAT se pueden construir formas complicadas rápida y fácilmente. Pruebe con ésta:

```
REPEAT 10 [CUADRADO RT 36 FD 25]
```

Cogito ergo LOGO
La tortuga se puede direccionar en términos de geometría cartesiana o de geometría "de tortuga". En la primera, las posiciones de la tortuga se expresan absolutamente: se miden desde los ejes X e Y imaginarios cuyo origen [0,0] es la posición CS; en geometría de tortuga, las instrucciones, relativas, se expresan en relación a la posición y el encabezamiento corrientes de la tortuga, sean cuales fueren



GEOMETRÍA CARTESIANA

GEOMETRÍA DE TORTUGA

Liz Dixon

Geometría de las coordenadas de la tortuga

La geometría de tortuga trata de las propiedades de las formas en sí mismas y no de la relación de formas respecto a un punto de referencia exterior, como es el caso en la geometría de coordenadas. La geometría de tortuga es relativa: los movimientos se realizan en una cantidad especificada de unidades a partir de la posición en curso en la pantalla. La geometría de coordenadas utiliza valores absolutos: la pantalla se imagina como una cuadrícula, con una cantidad definida de unidades que se extienden vertical y horizontalmente respecto al centro. Cada punto de la cuadrícula tiene un valor numérico específico y el movimiento se define en relación a estas referencias. Sin embargo, se puede utilizar la geometría de coordenadas con la tortuga.

Por ejemplo, la instrucción SETXY 20 30 desplaza



a la tortuga desde su posición corriente hasta el punto (20,30). Si se ha especificado PENDOWN, la tortuga trazará una línea a medida que se vaya moviendo; por el contrario, la instrucción PENUP hará que la tortuga se mueva sin dejar ninguna huella. El origen (0,0) está en el centro de la pantalla.

Los siguientes procedimientos realizan la misma tarea e ilustran la diferencia existente entre la geometría de tortuga y la geometría de coordenadas:

```
TO CUADRADO1
  REPEAT 4 [FD 50 RT 90]
END
```

```
TO CUADRADO2
  SETXY 0 50
  SETXY 50 50
  SETXY 50 0
  SETXY 0 0
END
```

Entrar CUADRADO1 o CUADRADO2 después de retornar a DRAW dará exactamente el mismo resultado. Pero ¿qué sucede si se desea rotar los dos cuadrados en 30º? RT 30 CUADRADO1 funciona correctamente en el primer caso, pero el segundo procedimiento ha de ser reescrito por entero (porque especifica coordenadas de pantalla absolutas), y ésta no es una tarea banal. Pero existen ocasiones en las cuales el empleo de coordenadas es útil: como observará a partir de estos ejemplos, SETXY es mucho más rápida para dibujar líneas que FORWARD.

Otra característica de la geometría de tortuga es su "cortedad de vista". La tortuga se preocupa de un solo movimiento por vez, va construyendo las formas dando una serie de "pasos" cortos. Juguemos a la tortuga y consideremos cómo se construye un círculo. Imagine que usted es la tortuga: ¿qué necesitaría para producir una forma circular? Se movería hacia adelante una corta distancia y giraría un poco, y repetiría esta secuencia muchas veces. En términos de LOGO esto se define así:

```
TO CIRCULO
REPEAT 360 [FD 1 RT 1]
END
```

Este procedimiento funciona, pero se ejecuta muy lentamente. Se puede lograr mayor rapidez si no se dibuja la tortuga en cada paso. La instrucción del LOGO HIDETURTLE (ocultar tortuga) se utiliza para hacer invisible la tortuga: SHOWTURTLE (mostrar tortuga) la dibuja otra vez. Nuestro ejemplo en realidad dibuja un polígono de 360 lados; las líneas que lo forman son tan cortas que dan la impresión de una curva uniforme. De hecho, 360 lados sirven para dar la ilusión de un círculo; un polígono de 36 ya bastaría. De modo que el siguiente procedimiento dibuja un círculo aceptable y lo hace a una velocidad mayor:

```
TO CIRCULO
  REPEAT 36 [FD 10 RT 10]
END
```

Ahora intente escribir procedimientos que produzcan un semicírculo, un cuarto de círculo y una sexta parte de círculo, y combine dos de estas formas para conseguir una forma de pétalo.

Volviendo a jugar a la tortuga otra vez, ¿cómo se movería usted si deseara una trayectoria en espiral hacia un punto, en vez de un recorrido en círculo alrededor del mismo? Todavía se movería una corta distancia antes de girar, pero en este caso giraría más y más cada vez (o recorrería una distancia más corta para cada giro). Este procedimiento de espiral es un poco largo, ya que lo hemos elaborado de modo que emplee sólo las instrucciones que hemos introducido hasta el momento, pero servirá para demostrar ese principio:

```
TO ESPIRAL
  FD 10 RT 10 FD 10 RT 20 FD 10 RT 30
  FD 10 RT 40 FD 10 RT 50 FD 10 RT 60
  FD 10 RT 70 FD 10 RT 80 FD 10 RT 90
END
```

Alan Coode



Probando el LOGO
El juego *Demolition turtle* (La tortuga demoledora), de Anthony Ginn, introduce la tortuga para el suelo y ayuda al niño a dominar las relaciones espaciales. Un jugador coloca una torre de juguete en algún lugar del tablero, y elige la posición y el encabezamiento de partida de la tortuga. El otro jugador programa la tortuga para que choque contra la torre, usando la menor cantidad posible de instrucciones

Complementos al LOGO

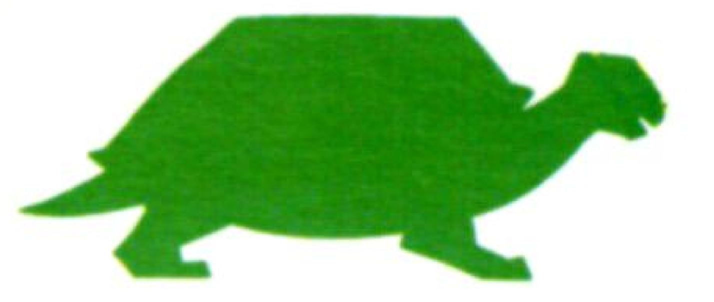
Los editores de LOGO son muy similares, pero cada uno posee sus peculiaridades debido al teclado de cada máquina concreta. Consulte el manual de LOGO para la suya.
En todas las versiones LCSI la instrucción para editar el procedimiento CUADRADO es EDIT "CUADRADO (comillas antes del nombre del procedimiento, pero no al final).
Para situar la tortuga en (20,30) utilice SETPOS [2 30] en todas las versiones LCSI.

Respuestas ejercicios

Triángulo	REPEAT 3 [FD 50 RT 120]
Pentágono	REPEAT 5 [FD 50 RT 72]
Hexágono	REPEAT 6 [FD 60 RT 60]
Rectáng.	REPEAT 2 [FD 25 RT 90 FD 50 RT 90]
Paralelog.	REPEAT 2 [FD 25 RT 70 FD 50 RT 110]
Rombo	REPEAT 2 [FD 50 RT 70 FD 50 RT 110]

Estrellas:

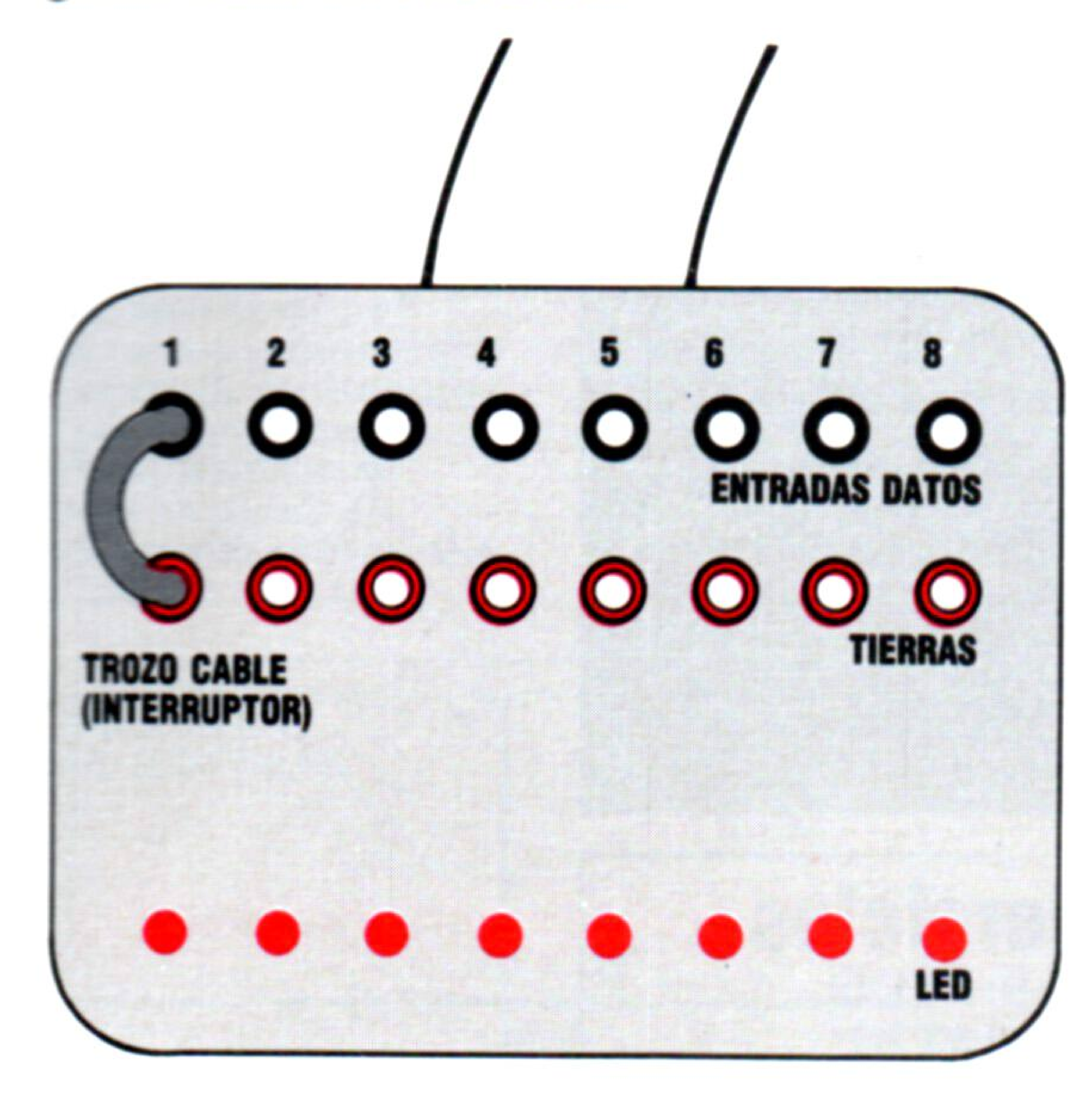
1. REPEAT 5 [FD 50 RT 144]
2. REPEAT 8 [FD 50 RT 135]
3. RT 30 REPEAT 3 [FD 50 RT 120]
   PU LT 30 FD 29 RT 90 PD
   REPEAT 3 [FD 50 RT 120]
4. REPEAT 10 [FD 50 RT 108]



Todo en su lugar

Hemos llegado al momento de probar cómo funciona la caja diseñada en los capítulos anteriores de este apartado

Una vez construida la caja buffer con sus LED, podemos escribir algún software sencillo para comprobar su funcionamiento como cronometrador de reacciones. Aquí utilizaremos el bit 7 del registro de datos para el interruptor de entrada y los bits del 0 al 6 para los LED. El objetivo consiste en medir el intervalo transcurrido entre el instante en que el ordenador enciende los LED y el momento en que se pulsa un interruptor. Primero debemos conectar un interruptor entre la terminal de entrada 7 de la caja y la correspondiente toma a tierra; este interruptor puede ser de cualquier tipo puesto que sólo se utilizará para abrir y cerrar el circuito entre el bit 7 del registro de datos y tierra. Como interruptor se puede emplear, por ejemplo, un trozo de cable flexible. Conecte el interruptor, asegurándose de que esté interrumpiendo el circuito entre el bit 7 y tierra (el valor del bit 7 debe ser uno). El siguiente diagrama ilustra las conexiones:



El programa de prueba primero inicializa el registro de dirección de datos de manera que el bit 7 se utilice para entrada y todos los otros para salida, y después coloca a cero el registro de datos. Al cabo de una demora aleatoria, se encienden siete de los LED y se pone en marcha el contador de tiempo interno del ordenador. El cronometraje continúa hasta que el bit 7 se envía abajo, cuando se realiza la conexión. Los LED están cableados de modo tal que se iluminan cuando hay un cero presente en el registro de datos y se apagan cuando hay uno.

9 VÍAS
RESISTENCIAS DE 240 OHM
RESISTENCIAS 4 K-OHM
CONECTOR POTENCIA
CONECTOR MINICON
CABLE DE LA PUERTA PARA EL USUARIO
E D7 D6 D5 D4 D3 D2 D1 D0 E
10 VÍAS
BUFFER HEXA
CONECTOR DIL
JUMPERS
DIODOS
PUENTE RECTIFIC.
CONDENSADOR DE 1 μF
E D0 D1 D2 D3 D4 D5 D6 D7 E
CABLE DE PUERTA PARA EL USUARIO
CORTE DE LAS PISTAS
Kevin Jones



LED

CONECTORES

Dentro de la caja

Pase un trozo de alambre estañado como muestra la ilustración a través de los contactos de los conectores. Suéldelo a cada uno de los contactos y compruebe la continuidad. Coja 20 cm de cable plano y separe tres cables, dejando un cable plano de nueve vías con la franja del borde de color. Pele y estañe los extremos de los cables. Suelde el cable de color en el conector cableado situado más a la izquierda. Ahora suelde los otros ocho cables restantes, por orden, en los contactos de los otros conectores. Pruebe la continuidad entre cada uno de los conectores y el final del cable al que estén conectados

ENLACE DE CABLES

DIODOS

CONDENS. ELECTROL. DE 1 μF

REGULADOR DE VOLTAJE

PUENTE RECTIFICADOR

+
– REGULADOR DE VOLTAJE

+
– CONDENSADOR ELECTROLÍTICO de 1 μF

Construcción de la placa

La placa ilustrada tiene 30 pistas con 45 agujeros y cabe exactamente en nuestra caja. Siga cuidadosamente las ilustraciones y no tendrá ningún problema para construir la placa. Utilice la menor cantidad posible de soldadura y tenga cuidado de no hacer puente entre las pistas; compruebe continuamente que esté colocando los componentes correctos en los lugares adecuados.
Los diodos, el condensador electrolítico, el puente rectificador y el regulador de voltaje se deben colocar todos en la dirección indicada; cualquier otra orientación los dañaría, de modo que estudie las marcas relativas a más y menos. Todos los componentes son sensibles al calor, por lo tanto no los "sobrecaliente" con el estaño.
Cuando instale el conector minicon y el de potencia procure colocar las patillas en los agujeros correctos de la placa, pero con cuidado de no torcer las patillas. Para los "cables de conexiones volantes" *(jumpers)* emplee los cables que quitó del cable plano.
Cuando todo esté colocado en su sitio en la placa, corte las pistas de cobre exactamente como muestra la ilustración. Para ello puede comprar una herramienta especial, o puede sostener una broca entre los dedos y hacerla girar en un agujero, cortando el cobre gradualmente. Suelde los cables planos a la placa. La orientación del conector y de los cables de los LED se indica mediante la franja de color, pero el cable de la puerta para el usuario exige un poco de atención; las dos líneas a tierra deben estar en los agujeros 1 y 10 (contando desde el borde de la placa) y las líneas de señal deben ir por orden en los agujeros del 2 al 9 de modo que la línea menos significativa esté más cerca del borde de la placa.
Por último, usando la placa como plantilla, corte ranuras en los lados de la caja para acomodar los conectores y el cable de la puerta para el usuario

```
10 REM * TIEMPO DE REACCION PARA EL BBC **
15 :
20 DDR=&FE62: REGDAT=&FE60
30 DDR=127: REM LINEAS 0-6 SALIDA
40 ?REGDAT=127: REM LED APAGADOS
50 :
60 CLS:PRINT "PREPARESE"
70 DEMORA=3000+RND(9000)
80 FOR I=1 TO DEMORA:NEXT:REM BUCLE DE DEMORA
85 FOR D=1 TO 200: NEXT D
90 :
97 REPEAT UNTIL ?REGDAT AND 128=1
100 ?FE60=0: REM ENCENDER LEDS
110 TIME=0: REM INICIALIZAR CONTADOR DE TIEMPO
120 REPEAT
130 UNTIL ?REGDAT AND 128=0: REM S/W ON
140 :
150 PRINT "TIEMPO INVERTIDO="TIME/100"SEG"
160 END
```

El programa emplea TIME, una variable reservada que devuelve un valor correspondiente a la cantidad de centésimas de segundo desde que TIME fuera puesta a cero por última vez. Haciendo uso del AND lógico en la línea 130 se aísla el bit 7 (valor 128) de modo que se pueda comprobar independientemente de los otros bits del registro de datos (véase p. 546). Cuando se conmuta el interruptor, el valor de un bit 7 pasa de uno a cero.

Se puede escribir un programa similar para el Commodore 64 utilizando su cronometrador interno, TI. Éste opera de manera distinta que TIME, devolviendo un valor en sesentavos de segundo desde que la máquina fuera encendida. Para emplearla debemos tomar el valor de TI al comienzo del intervalo a cronometrar y restarlo del valor de TI al acabar.

```
10 REM CMB 64 ** CRONOMETRADOR DEL TIEMPO DE REACCION **
20 :
30 DDR=56579: REGDAT=56577
40 POKE DDR,127: REM LINEAS 0-6 SALIDA
50 POKE REGDAT,127: REM LED APAGADOS
60 :
70 PRINT CHR$(147): REM LIMPIAR PANTALLA
80 PRINT "PREPARESE"
90 DE=3000+INT(9000*RND(1))
100 FOR N=1 TO DE:NEXT: REM BUCLE DE DEMORA
110 :
120 POKE REGDAT,0: REM ENCENDER LED
130 T=TI: REM TOMAR TIEMPO COMIENZO
140 IF PEEK(REGDAT) AND 128 <> 0 THEN 140
150 :
160 TM=(TI-T)/60: REM CALC INTERVALO
170 PRINT "TIEMPO INVERTIDO=";TM;"SEG"
180 END
```

En los próximos capítulos del curso analizaremos la construcción de salidas de corriente más alta, que son suficientes para activar motores eléctricos, y diseñaremos software para controlar motores bidireccionales y de velocidad variable.

Para probar el invento

1) Escriba un programa que encienda un LED cada vez en secuencia de izquierda a derecha.
2) Escriba un programa para hacer que los LED se iluminen en secuencia en las líneas del 0 al 6 (como en la pregunta 1), pero incluya un interruptor en la línea 7 para modificar la dirección de la secuencia. ¿Puede modificar su programa para que opere con un "tren" de tres LED?
3) Escriba un programa para simular el lanzamiento de un dado, utilizando seis LED y un interruptor.
4) Escriba un programa para simular la acción de un semáforo, utilizando tres LED.
5) Escriba un programa para contar la cantidad de "coches" (impulsos en un interruptor) que llegan mientras un semáforo está en rojo y para cambiar las luces cuando exceden de 10 o si ha transcurrido más de un minuto desde el último cambio.

Respuestas caja buffer

Éstas son las posibles soluciones a los Ejercicios de la página 1027:

Secuenciación de LED (I)

```
 10 REM CMB 64 VERSION 2.1
 20 DDR=56579: REGDAT=56577
 30 POKE DDR,225: REM TODAS SALIDA
 40 POKE REGDAT,225: REM TODOS LED APAGADOS
 50 GET A$
 60 FOR N=1 TO 7
 70 POKE REGDAT,255-(2^N)
 80 NEXT N
 90 IF A$="" THEN 50
100 END
```

```
 10 REM BBC VERSION 2.1
 20 DDR=&FE62:REGDAT=&FE60
 30 ?DDR=255: REM TODAS SALIDA
 40 ?REGDAT=255: REM TODOS LED APAGADOS
 50 REPEAT
 60 A$=INKEY$(1)
 70 FOR N=0 TO 7
 80 ?REGDAT=255-(2^N)
 85 FOR D=1 TO 200: NEXT D
 90 NEXT N
100 UNTIL A$ <> ""
110 END
```

Secuenciación de LED (II)

```
 10 REM CBM 64 VERSION 2.2
 20 DDR=56579: REGDAT=56577
 30 POKE DDR,127: REM L7 ENTRADA
 40 POKE REGDAT,255: REM TODOS LED APAGADOS
 50 GET A$
 60 FOR N=0 TO 7 STEP S
 70 POKE REGDAT,255-(2^N)
 80 NEXT N
 90 IF PEEK(REGDAT) AND 128=0 THEN S=-1
100 IF PEEK(REGDAT) AND 128=1 THEN S=1
110 IF A$="" THEN 50
120 END
```

```
 10 REM BBC VERSION 2.2
 20 DDR=&FE62:REGDAT=&FE60:S=1
 30 ?DDR=127: REM L7 ENTRADA
 40 ?REGDAT=255: REM TODOS LED APAGADOS
 50 REPEAT
 60 A$=INKEY$(1)
 70 FOR N=0 TO 7 STEP S
 80 ?REGDAT=255-(2^N)
 85 FOR D=1 TO 200: NEXT D
 90 IF REGDAT AND 128=0 THEN S=-1 ELSE S=1
100 NEXT N
110 UNTIL A$ <> ""
120 END
```

Secuenciación de LED (III)

Para el tren de tres LED, introduzca las siguientes modificaciones en la respuesta (2):

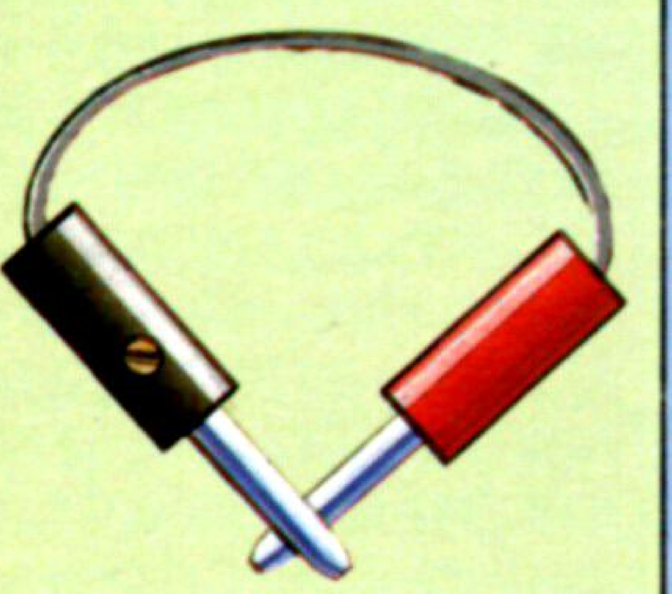
VERSION CBM

```
60 FOR N=2 TO 7 STEP S
70 POKE REGDAT,255-((2^N)+2^(N-1)+2^(N-2))
```

VERSION BBC

```
70 FOR N=2 TO 7 STEP S
80 ?REGDAT=255-((2^N)+2^(N-1)+2^(N-2))
```

CABLE DE PUENTE
Utilice 8 cm de cable aislado (una única línea del cable plano, por ejemplo) para conectar un enchufe rojo a un enchufe negro. Esto se denomina cable de parche y se utiliza para "puentear" un conector con otro. Haga ocho cables como éste



Kevin Jones

Lanzamiento de dados

```
 10 REM CBM 64 VERSION 2.3
 20 DDR=56579: REGDAT=56577
 30 POKE DDR,127: REM L7 ENTRADA
 40 POKE REGDAT,255: REM TODOS LED APAGADOS
 50 GET A$
 60 IF PEEK(REGDAT) AND 128 <> 0 THEN 60
 70 NR=INT(RND(1)*6)+1: REM SELECCIONAR NUM.
 80 POKE REGDAT,255-((2^NR)-1)
 90 IF PEEK(REGDAT) AND 128 <> 1 THEN 90
100 IF A$="" THEN 50
110 END
```

```
 10 REM BBC VERSION 2.3
 20 DDR=&FE62:REGDAT=&FE60:S=1
 30 ?DDR=127: REM L7 ENTRADA
 40 ?REGDAT=255: REM TODOS LED APAGADOS
 50 REPEAT
 60 A$=INKEY$(1)
 70 REPEAT UNTIL (?REGDAT AND 128)=0
 80 NUMERO=RND(6): REM SELECCIONAR NUMERO
 85 FOR D=1 TO 200: NEXT D
 90 ?REGDAT=255-((2^NUMERO)-1)
100 REPEAT UNTIL ?REGDAT AND 128=1
110 UNTIL A$ <>""
120 END
```

Semáforo (I)

```
 10 REM CBM 64 VERSION 2.4
 20 DDR=56579: REGDAT=56577
 30 POKE DDR,255: REM TODAS SALIDA
 40 POKE REGDAT,255: REM TODOS LED APAGADOS
 45 REM BIT2=ROJO,BIT1=AMBAR,BIT0=VERDE
 50 RJ=255-4: AM=255-2:VD=255-1
 60 GET A$
 70 POKE REGDAT,RJ
 80 FOR N=1 TO 200:NEXT:REM BUCLE DEMORA
 90 POKE REGDAT,RJ+AM
100 FOR N=1 TO 40:NEXT:REM BUCLE DEMORA
110 POKE REGDAT,VD
120 FOR N=1 TO 200:NEXT: REM BUCLE DEMORA
130 IF A$="" THEN 60
140 END
```

```
 10 REM BBC VERSION 2.4
 20 DDR=&FE62:REGDAT=&FE60:S=1
 30 ?DDR=255:REM TODAS SALIDA
 40 ?REGDAT=255: REM TODOS LED APAGADOS
 50 REM BIT2=ROJO, BIT1=AMBAR, BIT0=VERDE
 60 ROJO=255-4:AMBAR=255-2:VERDE=255-1
 70 REPEAT
 80 A$=INKEY$(100)
 85 FOR D=1 TO 200: NEXT D
 90 ?REGDAT=ROJO
100 FOR N=1 TO 200:NEXT:REM BUCLE DEMORA
110 ?REGDAT=ROJO+AMBAR
120 FOR N=1 TO 40:NEXT:REM BUCLE DEMORA
130 ?REGDAT=VERDE
140 FOR N=1 TO 200:NEXT:REM BUCLE DEMORA
150 UNTIL A$ <> ""
160 END
```

Semáforo (II)

Introduzca las siguientes modificaciones en la respuesta (4):

VERSION CBM

```
30 POKE DDR,127: REM L7 ENTRADA
65 T=TI: REM INICIALIZAR CRONOM.
75 IF PEEK(REGDAT) AND 128=0 THEN C=C+1
76 IF PEEK(REGDAT) AND 128 <> 1 THEN 76
77 IF C <=10 AND T-TI < 3600 THEN 75
```

VERSION BBC

```
30 ?DDR=127: REM L7 ENTRADA
75 T=0:C=0
95 REPEAT
96 IF ?REGDAT AND 128=0 THEN C=C+1
97 REPEAT UNTIL ?REGDAT AND 128=1
98 UNTIL T > 6000 OR C > 10
```

El "switch" o indicador

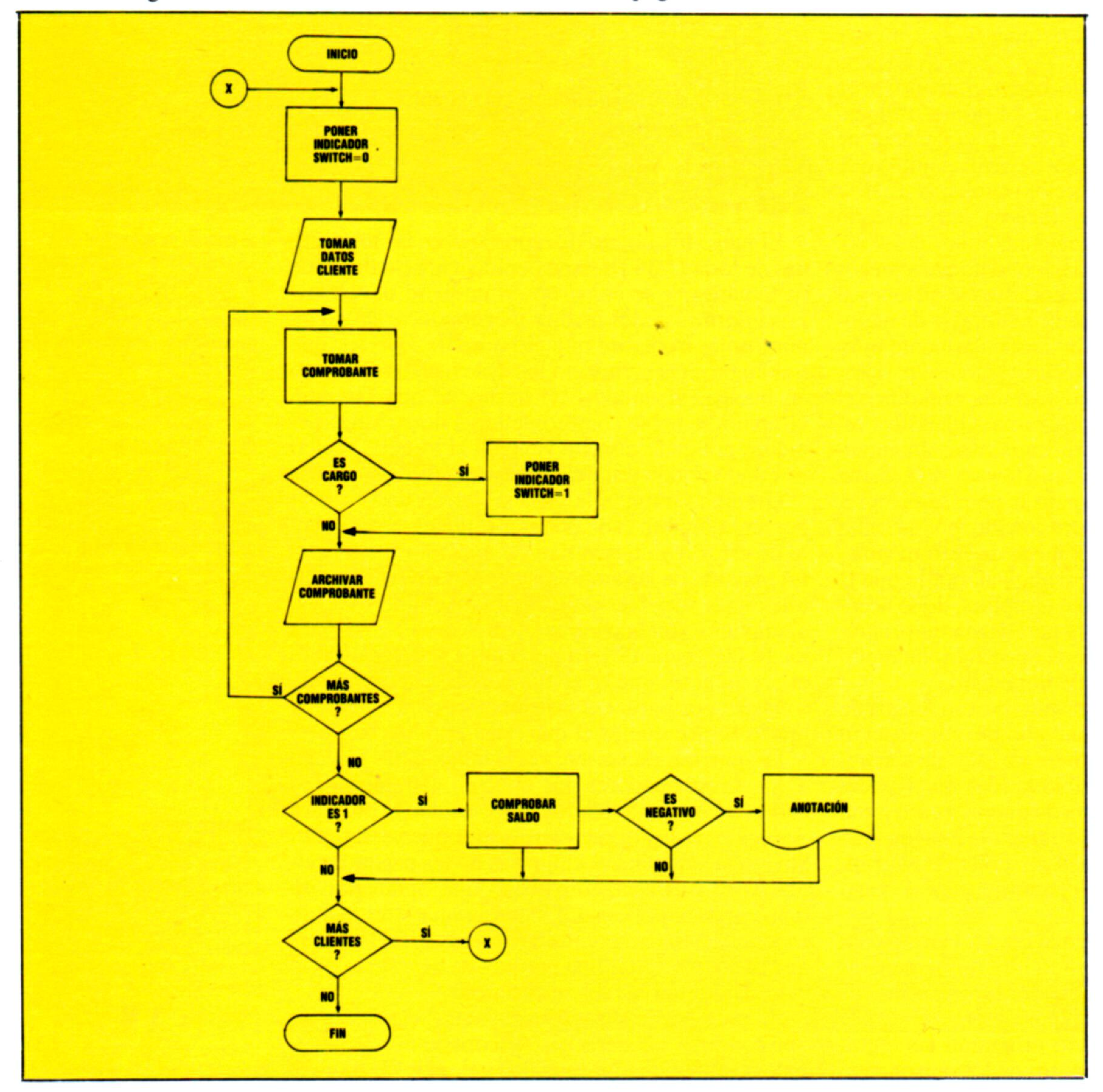
En esta ocasión nos referiremos a una variable de utilización esencial en el desarrollo de un programa: el "switch"

Un *switch* o indicador es una variable que puede asumir varios valores determinados, generalmente dos. Se utiliza en determinados puntos de un programa para indicar si el flujo del mismo ha pasado por un lugar concreto de la lógica o no; posteriormente, el mismo programa consulta el *switch* para saber por qué rama se ha llegado hasta el lugar donde se realiza la consulta.

En el ejemplo que presentamos se debe verificar el estado de cuentas de los clientes de un banco. Para ello hay que comprobar uno a uno los resguardos de operaciones de cada cliente. Cada comprobante representa una determinada operación de cargo o de abono. Una vez examinado cada resguardo, se archiva y se toma otro. Ahora bien, si entre los diferentes movimientos hay aunque sólo sea una operación de cargo, al acabar el proceso de todos los resguardos de ese cliente deberemos ver su estado actual, y en su caso señalar un posible saldo negativo antes de continuar con otro cliente.

En nuestro ejemplo, el *switch* actuará como una bandera indicadora de que el cliente tiene al menos una operación de cargo. Previamente al proceso de cada cliente, el *switch* se pondrá a valor 0. Este valor sólo se modificará si existe una operación de cargo, en cuyo caso tomará el valor 1. En la lógica del programa, una vez finalizado el proceso de todas las operaciones de un cliente, se consultará el estado de este indicador: si está todavía en su valor inicial significa que el cliente no tiene ninguna operación de cargo; si su valor es 1 significa lo contrario.

Obsérvese que, en general, un indicador sólo tiene dos estados, tal y como si se tratara de, por ejemplo, los dos estados de una bombilla, encendida o apagada.



En la cima del éxito

He aquí la historia de dos programadores adolescentes que se han hecho ricos a partir de la programación de una idea original

En el verano de 1984 Gremlin Graphics lanzó en Gran Bretaña un nuevo juego para ordenador llamado *Wanted: Monty Mole* (Se busca a Monty Mole). Se espera que el juego se convierta en un *bestseller*. De ser así, Peter Harrap, su programador, se encontrará con unas ganancias de varios miles de libras. Harrap tiene 19 años y *Wanted: Monty Mole* fue su primer juego comercial.

Peter Harrap pertenece a un pequeño grupo de "jóvenes genios" programadores, por lo general adolescentes, que pueden obtener considerables sumas de dinero si producen un juego que se coloque entre los diez mejores. Un muchachito de 14 años obtuvo casi 4 000 libras escribiendo software de juegos para el BBC Micro. Otro programador, de 17 años, fue obsequiado con un automóvil deportivo Lotus Esprit por la empresa de software para la cual trabajaba.

Al igual que la mayoría de los programadores de juegos, Harrap es autodidacta. El primer ordenador que tuvo fue un Sinclair ZX81 y, después de ver la reducida cantidad de software disponible que había en aquel entonces, decidió probar a escribir sus propios programas de juegos. Para hacerlo hubo de aprender el lenguaje máquina del Z80, puesto que el BASIC es demasiado lento para juegos recreativos. Harrap pronto dedicó sus atenciones al Sinclair Spectrum y se concentró en tratar de mejorar un juego que le resultaba particularmente interesante, el *Ant attack* de Quicksilva (véase p. 486). Harrap se metió en el código máquina y modificó el "paisaje" concebido en el programa. Quicksilva rechazó esta versión suya modificada, pero un amante local de los ordenadores, Ian Stewart, que estaba buscando un nuevo talento para su pequeña empresa de software, quedó impresionado por la cinta que Harrap le hizo llegar. El juego de Harrap violaba el copyright del *Ant attack* original, por lo que la empresa de Stewart, Gremlin Graphics, no lo aceptó. Sin embargo, la empresa necesitaba un programador muy cualificado para el Spectrum, de modo que contrataron a Harrap de inmediato.

Todas las empresas de software buscan juegos de temática interesante e inusual con los que esperan trepar hasta el primer puesto de las listas de software. Gremlin Graphics concibió la idea de *Wanted: Monty Mole* con la esperanza de que su tópico tema de la huelga de los mineros atrapara la imaginación del público. La idea la puso en práctica Harrap, quien escribió un programa para ejecutar el juego en el Sinclair Spectrum.

La programación del juego le llevó a Harrap cuatro meses. Empezó escribiendo una rutina generadora de sprites para poder diseñar rápidamente las diversas formas de los gráficos, luego dibujó la mina de carbón y por último programó las reglas que determinan los movimientos de los personajes.



Ian McKinnell

El detalle de los detallistas
La presentación del punto de venta tal vez sea la más importante de todas las interfaces para el usuario; aquí se combinan los efectos de la publicidad, los reportajes, el empaquetamiento, la comercialización y el atractivo del producto para influir en la decisión del cliente. La disponibilidad es, obviamente, un factor crucial para las ventas, de modo que conseguir que el producto se distribuya a través de cadenas de tiendas al detall es uno de los objetivos básicos de todos los editores de software

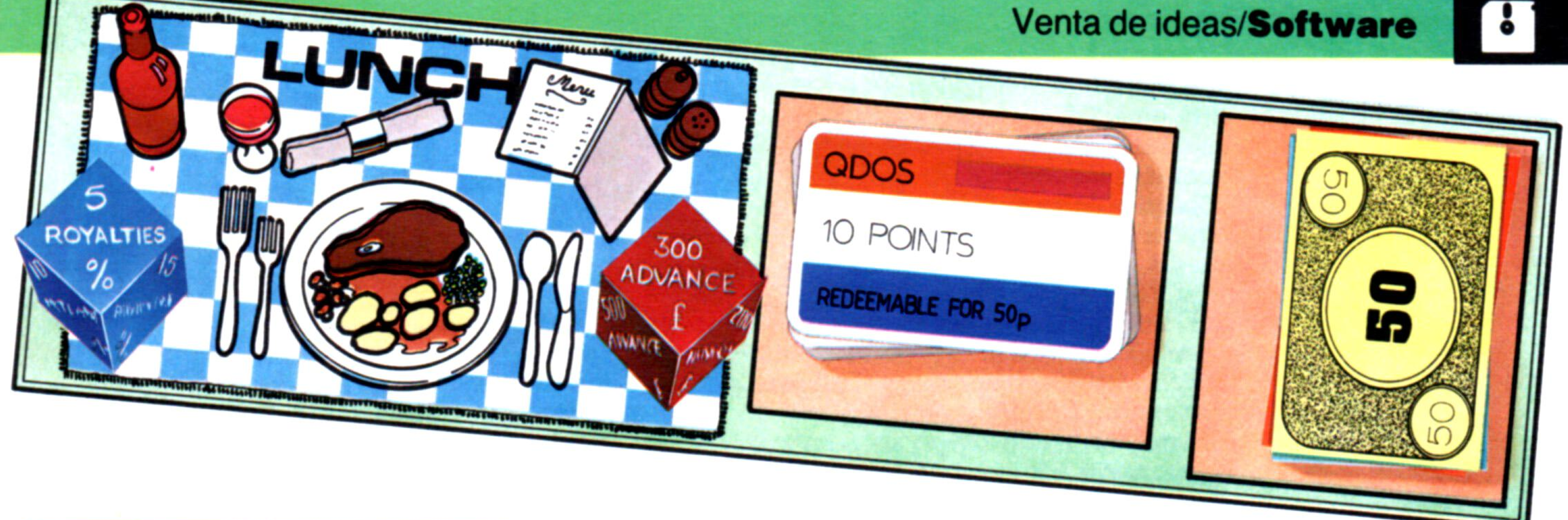
El juego fue objeto de reportajes en las televisiones de toda Gran Bretaña debido a la actualidad de su temática, y se convirtió en un éxito de ventas. Los informes de los medios de comunicación no ignoraron la ironía de que el padre de Harrap, que trabajaba en la carbonera local, se hallaba él mismo en huelga, ni tampoco el hecho de que Gremlin Graphics se había comprometido a donar cinco peniques por cada cassette vendida al Miners' Welfare Fund (Fondo para el bienestar de los mineros).

Gremlin Graphics le paga a Harrap un *royalty* o porcentaje sobre cada cassette vendida. El montante de estos royalties varían de una empresa a otra. La mayoría de las empresas abonan royalties según una escala variable: el 5 % para los primeros 3 000 ejemplares vendidos, por ejemplo, y el 7 % para los 10 000 siguientes. El porcentaje se calcula sobre el precio neto, que es el que percibe la empresa de software por parte del distribuidor, o bien sobre el precio bruto, que es el que paga el público.

La mayoría de las empresas también les abonan a los programadores un *anticipo*. El mismo se abona a cuenta de los royalties, es decir, que será deducido de los primeros royalties que se abonen. Sin embargo, muchas empresas no les pagan royalties a sus programadores, sino que optan por un único pago al principio. Los programadores han de sopesar detenidamente esta clase de arreglos, porque pueden resultar una pérdida de beneficios en el caso de que las ventas sean elevadas.

En un acuerdo basado en royalties, el programador conserva los derechos de copyright del juego. Esto significa que en caso de cualquier litigio es el

Consiguiéndolo
Se cuentan por miles los usuarios de micros personales que están intentando escribir paquetes de software; algunos de ellos llegan realmente a alcanzar el éxito, encontrando un editor y ganándose algún dinero. Puede que nuestro juego imaginario exagere un poco el elemento de cambio en el proceso, pero con toda seguridad, es una lotería...



PILLAJE

El juego de Juegos

- Se estimula a los jugadores para que engañen y estafen a los otros y para que infrinjan la mayor cantidad de reglas posible.
- El objetivo del juego consiste en echar a los demás jugadores del mismo mediante la acumulación de todos los QDOS y el dinero. Esto se consigue llevándole los detalles de su IDEA a un EDITOR (PUBLISHER) y llegando a un acuerdo durante el LUNCH. Su primer acuerdo sólo le reporta un royalty; los acuerdos subsiguientes le reportan un royalty y un anticipo inmediato.
- Una vez hecho el acuerdo, su cassette sale a la venta en un BUGMART, obteniendo beneficios abonados por todos los jugadores que aterricen allí.
- Cada vez que pulse RETURN, usted obtiene una nueva IDEA y recoge sus royalties, que se calculan multiplicando sus QDOS por su porcentaje de royalty.
- Cuando caiga en el teclado (GOTO KEYBOARD), debe ir directamente; no debe pulsar RETURN, ni recoger sus ganancias, y debe quedarse allí hasta que le salga un seis o consiga robar la IDEA de alguien.
- BERGSTROM toma su último juego y lo empaqueta con su nuevo micro. Ello hace que sus ingresos por ese juego se dupliquen.

Steve Cross

programador el demandante, o bien el demandado. Si el programador recibe por su juego una única cantidad global, la empresa de software por lo general también comprará el copyright.

Además de los royalties, Harrap también percibe una cantidad fija, pero tiene un contrato exclusivo con Gremlin Graphics por un año y no puede trabajar para ninguna otra firma de software. Muchos contratos también incluyen una cláusula de opción en virtud de la cual la empresa de software tiene el derecho exclusivo de preferencia para aceptar o rechazar cualquier idea o programa.

Para que un juego tenga éxito, una empresa persigue unas ventas de al menos 15 000 ejemplares durante su vida, que suele ser de cuatro o cinco meses. En el caso de *Wanted: Monty Mole*, la empresa inicialmente reprodujo 10 000 ejemplares para cada máquina (Spectrum y Commodore 64), con la opción de reproducir más. Cuando se lanzó el juego, Ian Stewart predijo unas ventas de 20 000 para cada máquina y afirmó: "Creo que definitivamente se convertirá en el número uno".

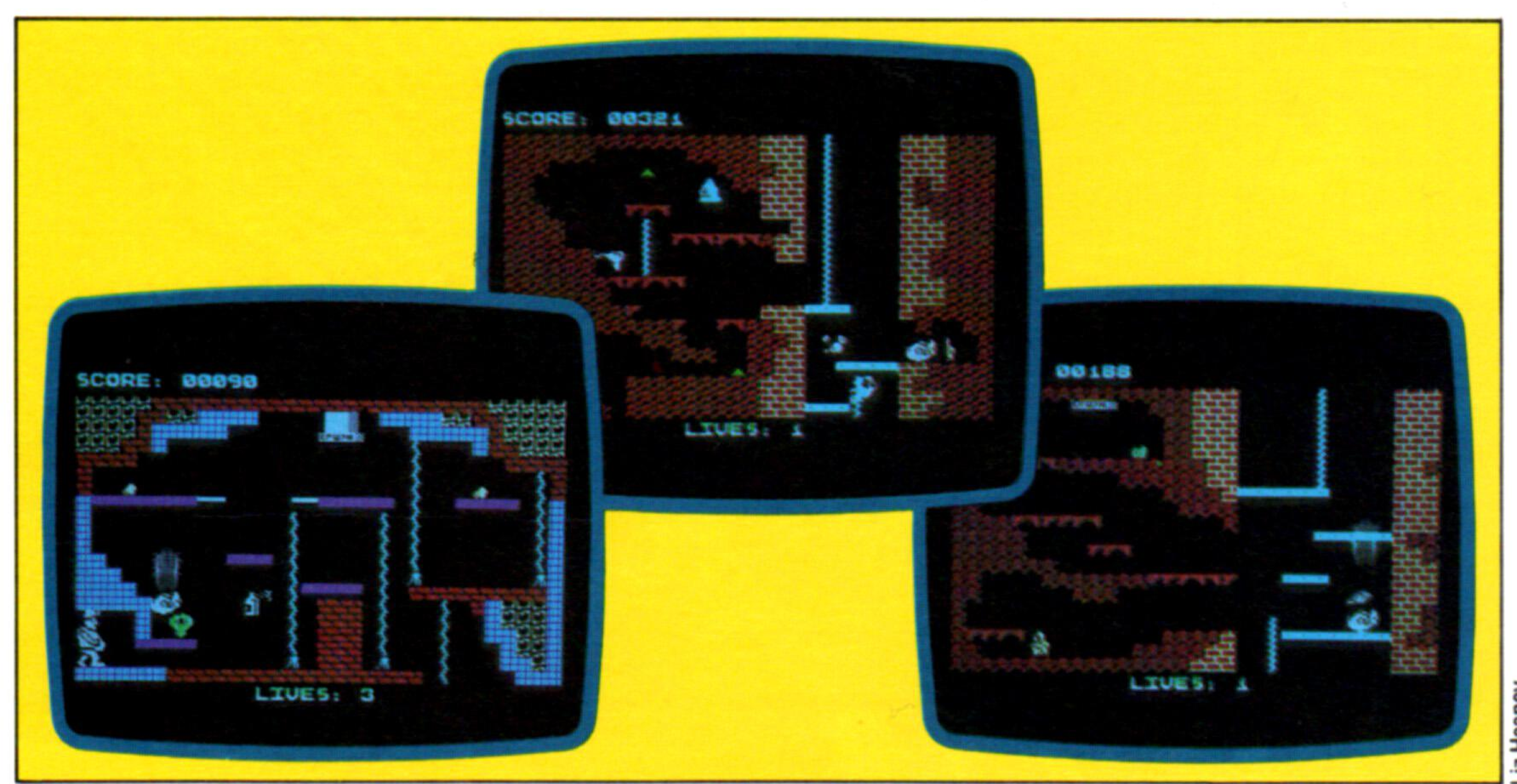
Christopher Kerry es otro programador a quien el éxito le ha sonreído incluso antes. A los 17 años abandonó los estudios para trabajar por libre para la empresa de software House of Thor. Kerry escribió el juego *Jack and the Beanstalk* en el ordenador Spectrum que tenía en su casa de Sheffield. Sin publicidad, y después de haber estado a la venta durante apenas dos meses, se habían vendido "decenas de miles", según la empresa (que no estaba preparada para revelar la cifra exacta). En realidad House of Thor tenía una actitud muy protectora hacia Kerry, aduciendo que temían que alguna otra empresa estuviera tratando de "pescarlo". Además de Christopher Kerry, House of Thor utiliza a unos 15 programadores, la mayoría adolescentes.

Al igual que todas las empresas de software, House of Thor acepta de buen grado los programas que le envía la gente para su evaluación y, si parecen prometedores, la empresa se pone en contacto con el programador ese mismo día. Una respuesta rápida es de vital importancia, porque bien podría ser que el mismo software se le hubiera ofrecido también a otras empresas. House of Thor sólo se interesa por juegos de estilo recreativo y éstos sólo se pueden escribir realmente en lenguaje máquina. La empresa también busca temas originales y, por lo tanto, no toma en cuenta aquellos programas que sean prácticamente copias de juegos ya existentes: uno no puede limitarse a modificar los gráficos o cambiarles los nombres a los monstruos, por ejemplo. Aparte de las posibles acciones legales que ello podría suponer, es poco probable que esta clase de juegos se venda bien.

Salamander es una gran firma de software que tiene contratados alrededor de 40 programadores. Chris Holland, su director gerente, les hace sugerencias a los potenciales escritores de software: "Cuando uno puede hacer realmente el agosto es escribiendo software para las máquinas nuevas apenas éstas salen al mercado. Nosotros recibimos de buen grado software para los micros Amstrad, Atmos y los MSX. Con uno como el Spectrum ya es más difícil, a menos que se trate de una idea completamente nueva". La empresa posee una gama de alrededor de 50 títulos para la mayoría de los ordenadores personales populares, no tan sólo para el Commodore 64 y el Spectrum.

En vez de intentar venderles los programas a las firmas de software, hay algunos programadores que crean sus propias empresas. Llamasoft y Bug-Byte son apenas dos ejemplos de algunas casas de software que han sido fundadas por jóvenes programadores de juegos. Lamentablemente, se necesita un capital bastante grande para empezar. La empresa de Ian Stewart, Gremlin Graphics, empezó con un solo juego, *Potty Pigeon*, pero pronto esperan contar con una gama de cinco programas. Los cuatro directores de la empresa han puesto dinero de su propio bolsillo para pagar la publicidad y la distribución; un anuncio en color de una página en una revista de ordenadores vale más de mil libras si se incluyen los costos de producción. Ni siquiera una gama de juegos de éxito constituye una garantía de seguridad financiera. Imagine (véase p. 559) se declaró en quiebra en julio de 1984 debiendo más de 1 millón de libras, a pesar de que en un momento dado la empresa estuvo vendiendo software de juegos por valor de 300 000 libras al mes.

Montar una empresa de software puede ser arriesgado; además de buenos juegos, también se necesita asesoramiento financiero. Pero cualquiera puede enviarle un programa a una empresa de software para ver "qué pasa". Quién sabe, tal vez usted haya escrito el próximo número uno en ventas.



Los pozos
El programador que escribió *Wanted: Monty Mole* es el arquetipo del genio de software: joven, autodidacta, brillante y bien remunerado. El juego en sí mismo dista mucho de ser un caso típico; adopta una posición petulantemente antisindical respecto a la huelga de los mineros de 1984, pero cada ejemplar que se venda supondrá la donación por parte de los editores de cinco peniques para el Miners' Welfare Fund (Fondo para el bienestar de los mineros)

Liz Heaney



Interface sonora

En esta ocasión analizaremos parte del hardware y del software que permiten al usuario utilizar la interface MIDI

El Micon, o *MI*DI *con*troller (controlador de MIDI), lo produce XRI Systems y es bastante representativo de los paquetes MIDI que producen las casas de software más modestas. Está diseñado para utilizarse con el Sinclair Spectrum, y se compone de una interface MIDI y una cassette de software. Su facilidad de secuenciador proporciona ocho pistas separadas, cada una de ellas con capacidad para 2 951 eventos (notas, pausas, etc.). La música se entra en el teclado del sintetizador conectado en interface y se utiliza la tecla Space del Spectrum para definir la cantidad de eventos a ejecutar.

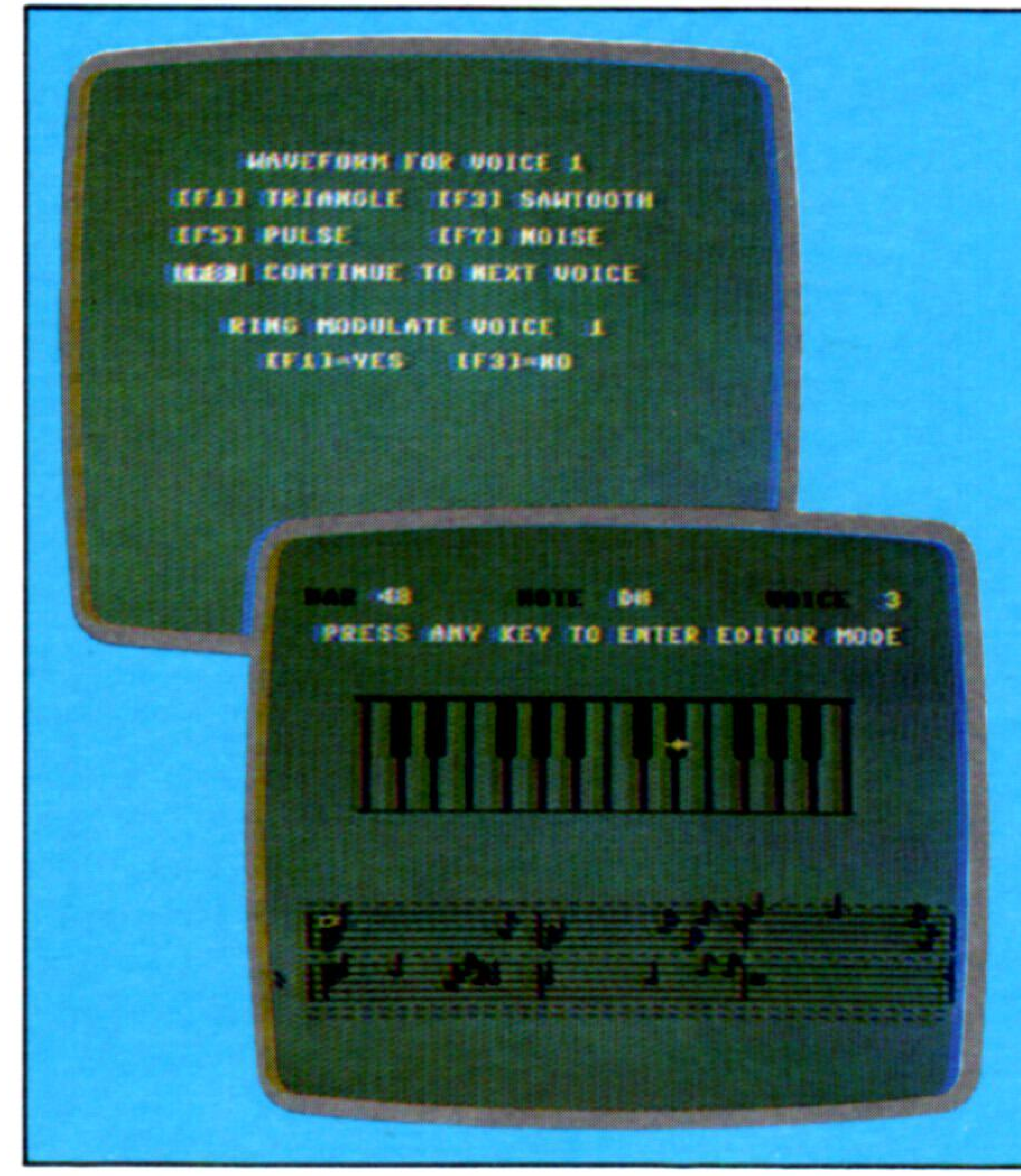
Mientras se va utilizando el sintetizador, la música se va desarrollando en la pantalla en una forma tosca de notación estándar con pentagrama de cinco líneas. Ésta proporciona la clave de *fa* o de *sol*, la gama completa de duraciones (incluyendo notas puntuadas), sostenidos y bemoles, e indicadores de staccato (notas muy cortas). La notación es pobre para indicar pausas (aquellas duraciones donde *no* se producen sonidos de notas juegan un papel esencial en cualquier composición musical) y los rabos de las notas apuntan siempre hacia arriba, no hacia arriba o hacia abajo según el lugar ocupado en el pentagrama. Además, las frases de notas cortas no se reorganizan de forma racional en grupos que reflejan el tiempo o ritmo implícito de la música. El resultado de todo ello es difícil de leer, excepto como guía aproximada, pero se puede imprimir mediante la impresora ZX.

La utilidad de la visualización se hace evidente cuando se trata de editar la composición. La música se divide en compases y, entrando un número de compás, se pueden eliminar, insertar o modificar notas individuales especificando la nueva nota. Se pueden borrar compases enteros y se pueden repetir agrupamientos de compases en cualquier lugar dentro de la secuencia. Es posible guardar en cinta hasta 10 secuencias (casi 24 000 notas), junto con las características apropiadas para su reproducción. La velocidad de la reproducción se puede definir dentro de cuatro milisegundos o controlar con una máquina de ritmos utilizando un conector SYNC IN en la interface.

El Micon también secuencia en tiempo real. En efecto, "escuchará" una ejecución en un sintetizador compatible con la MIDI y entrará todos los datos en la memoria del Spectrum. No visualizará una notación de la ejecución, pero es útil en otros dos sentidos. En primer lugar, permite que los músicos de teclado oigan una grabación de su propia actuación, sin tener que preocuparse por establecer niveles de entrada en cinta; su instantaneidad será especialmente valiosa para la educación musical. En segundo lugar, siempre que la música requerida esté "bajo los dedos" del ejecutante, alivia gran parte del tedio de la secuenciación de golpes de compás. No obstante, el ejecutante aún necesita de un metrónomo o patrón de máquina de ritmos junto al cual tocar, con el fin de llevar un tempo constante, porque de lo contrario la frase se reproducirá con todos los errores de ritmo del ejecutante. Al igual que con cualquier otro sistema MIDI, los resultados obtenidos con el Micon dependerán de la sofisticación del diseño del sintetizador con el que esté conectado.

El paquete MIDI de Jellinghaus Music System vale un poco menos que el Micon y además está diseñado para utilizarlo con una gama más amplia de máquinas: el Apple II, el Commodore 64 y el Sinclair Spectrum. Básicamente es similar al Micon en cuanto a que la información musical se entra desde el teclado del sintetizador conectado en interface. La principal diferencia es que las frases sólo se pueden entrar en tiempo real y que la visualización en pantalla de la música no responde a la notación estándar de pentagrama de cinco líneas.

La organización de la música en compases se realiza especificando los tiempos, es decir, cuántas negras o corcheas entran en cada compás. Hay cuatro tipos de compás disponibles: tres, cuatro o cinco negras por compás (3/4, 4/4 o 5/4 tiempo), o siete corcheas (7/8) y, al igual que con el Micon, el compás de la ejecución se ha de coordinar mediante un metrónomo separado o una máquina de ritmos. Si la actuación en tiempo real se desvía en velocidad, entonces los datos de entrada se convertirán en una cadena de notas "abiertas", sin compás. Después de que la frase esté lista para su reproducción, no habrá ningún problema hasta que no se produzca algún agrupamiento que no sea de 3/4, de 4/4, de



Control de sonido

Aprovechando el chip de sonido del Commodore 64, se puede escribir un programa en BASIC para entrar música con un formato visual mediante una palanca de mando y hacer que la misma se reproduzca. En el ejemplo que vemos aquí aparece una luz en un teclado gráfico y se puede llevar hacia arriba de la escala pulsando *forward* en la palanca de mando, o hacia abajo de la escala tirando hacia atrás. Cuando uno llega a la nota deseada, mueve la palanca de lado a lado para seleccionar la duración de la nota. Ésta se "graba" cuando se acciona el pulsador de disparo, y se está entonces preparado para introducir la siguiente nota de una frase

Liz Heaney



1 Control por ordenador
El Commodore 64 proporciona gestión de memoria y operación conducida por menú del sistema de música a través de software. Se puede utilizar la unidad de disco, o la grabadora de cassette, para efectuar grabaciones digitales de frases y almacenarlas para su uso ulterior

5/4 ni de 7/8. Apenas sucede esto, falla la facilidad de secuenciación. Esto también significa que no se puede entrar un tiempo como 12/8, aunque la velocidad sea perfecta, una omisión lamentable, puesto que 12/8 es un tiempo de *swing* comúnmente utilizado en jazz, rock, funk y reggae.

Siempre y cuando se haya dado entrada a una frase correctamente, los datos se visualizan en la pantalla en columnas, mostrando la octava en la cual se produce el evento de nota, la altura de la nota dentro de la octava, la duración en términos musicales (negras y corcheas), *gate-time* (un parámetro que se utiliza para otorgar un cierto fraseado a la línea musical) y los valores de "sensación táctil" del teclado, de 0 a 9.

Este tipo de visualización de hecho proporciona más información sobre una base de nota por nota que el empleo de la notación estándar que utiliza el Micon, que jamás ha incorporado ni *gate-time* ni niveles de velocidad, al menos no en forma numérica. Ello se debe a que la capacidad para especificar tales parámetros se ha introducido en el vocabulario musical sólo a raíz del advenimiento de la electrónica.

No obstante, muchos músicos sin ninguna clase de entrenamiento formal desarrollan enseguida una habilidad para mirar una sección de notación estándar y, mediante el empleo de la forma "subida y bajada" de las notas esbozada en el pentagrama, se hacen alguna idea sobre cómo sonará una melodía, o qué grupo de notas de un acorde se suman armónicamente. Esto significa que, en la práctica, la notación se puede leer, y una frase musical "escuchar" mentalmente en tiempo real. Las visualizaciones de columnas pueden ser excelentes para comprobar datos, pero son pocas las personas que pueden tararear una melodía a partir de ellas. Un sistema ideal incluiría los dos tipos de visualización; pero, al no ser éste el caso, la torpe versión del Micon de la notación estándar es más eficaz e idónea para desarrollar aptitudes musicales.

Sintetizadores conceptuales

Puede que el principal problema de cara al usuario de un ordenador personal con inclinaciones musicales no sea el de hallar el paquete MIDI correcto, sino el de obtener un sintetizador compatible con la MIDI que sea suficientemente sofisticado como para hacer el mejor uso de la interface sin que por ello el usuario necesite un segundo préstamo para pagarlo. Si exigimos que el sintetizador sea capaz de manipular el 50 % de las pistas de refuerzo de un disco *pop* sencillo típico de hoy, nos encontramos con que uno de tales instrumentos cuesta entre 200 000 y 300 000 pesetas. Por debajo de esta gama de precios, la mayoría de los sintetizadores son sumamente limitados en cuanto a su aplicación.

Una posibilidad a tener en cuenta es lo que se ha dado en llamar *sintetizador conceptual*: típicamente, un paquete de software avanzado y un teclado periférico que utiliza el potencial para generación de sonido de un microordenador, en vez de una interface a un sintetizador que es probable se haya diseñado antes de que se desarrollara la MIDI.

Un ejemplo es el PDSG (*Programmable Digital Sound Generator:* generador digital y programable de sonido). El mismo se conecta en interface con el microordenador BBC Modelo B. Incluye un teclado musical sensible a la velocidad, de 61 notas y dos pedales de pie. El teclado sensible representa la di-



2 Six-Trak, de Sequential Circuits
Potente sintetizador con una grabadora de cinta de pistas múltiples incorporada.
El Six-Trak (seis pistas) se llama así porque puede grabar en seis pistas.
A diferencia de muchos sintetizadores de esta clase, permite que cada voz musical tenga un sonido diferente, lo que posibilita la creación de sonidos complejos. Cada voz se puede controlar por frecuencia, timbre, forma de onda, ligadura de frecuencia y flexión de nota

3 Model 64, de Sequential Circuits
Enchufando el Model 64 a la puerta para ampliación de memoria del Commodore 64 es posible incorporar un sintetizador equipado con MIDI y la memoria, almacenamiento en disco o cassette y visualización en video del ordenador a un sistema musical. El Model 64 almacena información de compás, altura y modulación para hasta 4 000 notas en modalidad de grabación. Para la reproducción, la interface puede enviar la señal digital exactamente tal como fue recibida desde el teclado (tiempo real), o corregirla para una indicación determinada de tiempos (golpes de compás)



Ian McKinnell

ferencia entre un tacto sensible y uno "muerto" en la ejecución, y los datos de sensibilidad de la ejecución en tiempo real se pueden almacenar para grabación y reproducción. Posee 32 unidades generadoras de sonido, en lugar de osciladores, cada una de las cuales puede tener hasta 11 características definidas. Si así se requiere, se pueden utilizar las 32 unidades para producir una única nota. Esta facilidad sola, en manos de un usuario capaz, le confiere al PDSG una riqueza y una variedad de sonido al nivel de la mayoría de los sintetizadores que hemos reseñado en el recuadro.

Si se le asigna una sola unidad generadora a cada nota, entonces una frase programada puede constar de 32 líneas individuales. Por otra parte, se puede utilizar cierta proporción de generadores para material secuenciado, y tocar los restantes en tiempo real junto a la frase. Las características de forma de onda se visualizan en la pantalla, ofreciendo la oportunidad de analizar los sonidos visualmente, un apoyo inestimable para adivinación auditiva, y un factor gracias al cual el PDSG resulta extraordinariamente apropiado para la educación musical.

La principal desventaja del PDSG es su representación del sonido en la fase de conversión de digital a analógico. El oído y el cerebro humanos pueden interpretar sonidos a través de una anchura de banda entre 20 Hz y 20 KHz. Los sonidos naturales, incluyendo aquellos producidos por instrumentos musicales acústicos, son activos dentro del total de esta anchura de banda y más. El PDSG, sin embargo, puede representar sonido sólo dentro de una anchura de banda de hasta 12 KHz. A consecuencia de ello, su calidad de sonido es comparable con la de un sistema de alta fidelidad doméstico adecuado, y los fabricantes dan por sentado que para completar el sistema se utilizarán altavoces y un amplificador *hi-fi* domésticos. La mayoría de los músicos sin sintetizador quedarían consternados si ésta fuera su única opción de amplificación. La firma británica Clef Products tiene planificadas versiones del generador digital y programable de sonido (PDSG) para conectar en interface con otros microordenadores.

La MIDI se ha considerado como una brecha porque les proporciona a los usuarios de microordenadores acceso a verdaderos sintetizadores musicales. El sistema PDSG es suficientemente avanzado en muchos aspectos como para que los propietarios de sintetizadores consideren, en cambio, la posibilidad de comprarse un microordenador y un "sintetizador conceptual".

Colaboración a la música

El desarrollo de la interface MIDI les proporciona a los usuarios de microordenadores una gama de posibilidades para hacer música. Pero, al mismo tiempo, existe el riesgo de adquirir un paquete caro (la interface propiamente dicha, el software adicional y un sintetizador) sólo para quedar abrumado por los intrincados detalles del sistema. Una alternativa es comenzar con un sistema de música económico para familiarizarse con los fundamentos de la música electrónica. Tal sistema, por supuesto, debe ser suficientemente bueno como para que resulte satisfactorio desde el punto de vista musical y despierte el interés por las demás posibilidades de la música electrónica. Un buen "paquete de iniciación" es el Ultisynth 64, basado en cassette y producido por Quicksilva. Éste explota el chip SID *(sound interface device)* del Commodore 64 y sus tres osciladores.
Utilizando el paquete, cada una de las teclas del teclado del Commodore se convierte en un control independiente para generar y definir sonido. Están disponibles las cuatro formas de onda básicas (sinusoidal, cuadrada, triangular y aserrada), junto con la posibilidad de establecer incrementos para definir las características de ataque-sostenimiento-decaimiento-liberación (ADSR o envoltura). Los sonidos se pueden filtrar (es decir, se puede restar de la salida una anchura de banda de frecuencias especificada) para caracterizar al sonido aún más. La modulación de anillo (un proceso que proporciona la suma y la diferencia de dos frecuencias cualesquiera) también se incluye. Esto es útil para producir sonidos tipo campana bastante auténticos. Además se puede escribir el ritmo, incorporar ritmos preestablecidos y secuenciar 2 048 notas.
Las facilidades del Ultisynth se asemejan mucho a las del VCS 3, un sintetizador "clásico", controlado por voltaje, de finales de los sesenta.
Otro sistema basado en cassette adecuado para el principiante es el Multisound de Romik, que es muy parecido al Ultisynth en cuanto a sus facilidades de control, pero que ofrece la visualización de un teclado musical. En este teclado las posiciones se seleccionan utilizando un cursor y las notas se definen con información entrada mediante el teclado alfanumérico del ordenador.

Opciones de elección

En el tema de guiar al usuario en un programa veremos aquí dos técnicas típicas: el sistema de menús y el sistema de comandos

Los menús pueden ser simples listas de ítems numerados o pueden ser pantallas llenas de elaborados iconos, pero el principio que rige su utilización es siempre el mismo. El menú se emplea cuando el programa alcanza, en su lógica, una bifurcación de caminos múltiples; entonces se solicita al usuario que elija qué ruta tomar a partir de una lista de las opciones disponibles que se visualiza en la pantalla. Los programas basados en menús tienden a asumir estructuras arborescentes: el usuario entra en el árbol por la "raíz" y es guiado mediante las opciones del menú hacia una de las "hojas", donde se halla la información o función.

La principal ventaja de este enfoque es que el usuario necesita poco o ningún conocimiento acerca de la estructura del programa, porque el camino está bien "señalizado" a través de toda la ruta. No obstante, a los usuarios más experimentados la labor de ir abriéndose paso a través de una cadena de menús les resulta tediosa cuando se trata de tareas que se repiten con frecuencia. Los novatos, asimismo, pueden tener dificultades con la estructura arbórea; corregir una decisión errónea implica ir trabajando hacia atrás a través de todos los menús, hasta el punto en el cual se incurrió en el error, volver a entrar la opción correcta y después seguir adelante a partir de allí. El Prestel es una estructura particularmente "profunda" de este tipo y los usuarios se encuentran frecuentemente con este problema. No es en absoluto necesario que los menús conformen un árbol; se pueden organizar en red mediante la utilización de bucles.

Diseñar un sistema de menú puede ser difícil, a pesar de que la programación propiamente dicha es relativamente fácil. El principal problema es que se debe especificar claramente todo el programa completo antes de escribir ninguna instrucción. (Ésta es, de todas maneras, una buena costumbre, pero no siempre es fácil.) Agregar nuevas funciones en una etapa ulterior puede implicar la modificación de varios menús anteriores del programa, y esto puede requerir una importante reestructuración. Cuando se diseña el programa se debe incluir toda la lógica de los menús en una única rutina que llame a las rutinas de las "hojas" cuando se llegue a las mismas. La rutina de menú se puede considerar, por consiguiente, como una forma más compleja de la rutina de control normal, con todas las bifurcaciones internas controladas por el usuario. Ello produce un diseño pulcro y sirve para separar la lógica de control de las partes funcionales del programa, permitiendo desarrollar y depurar cada una de ellas de forma independiente.

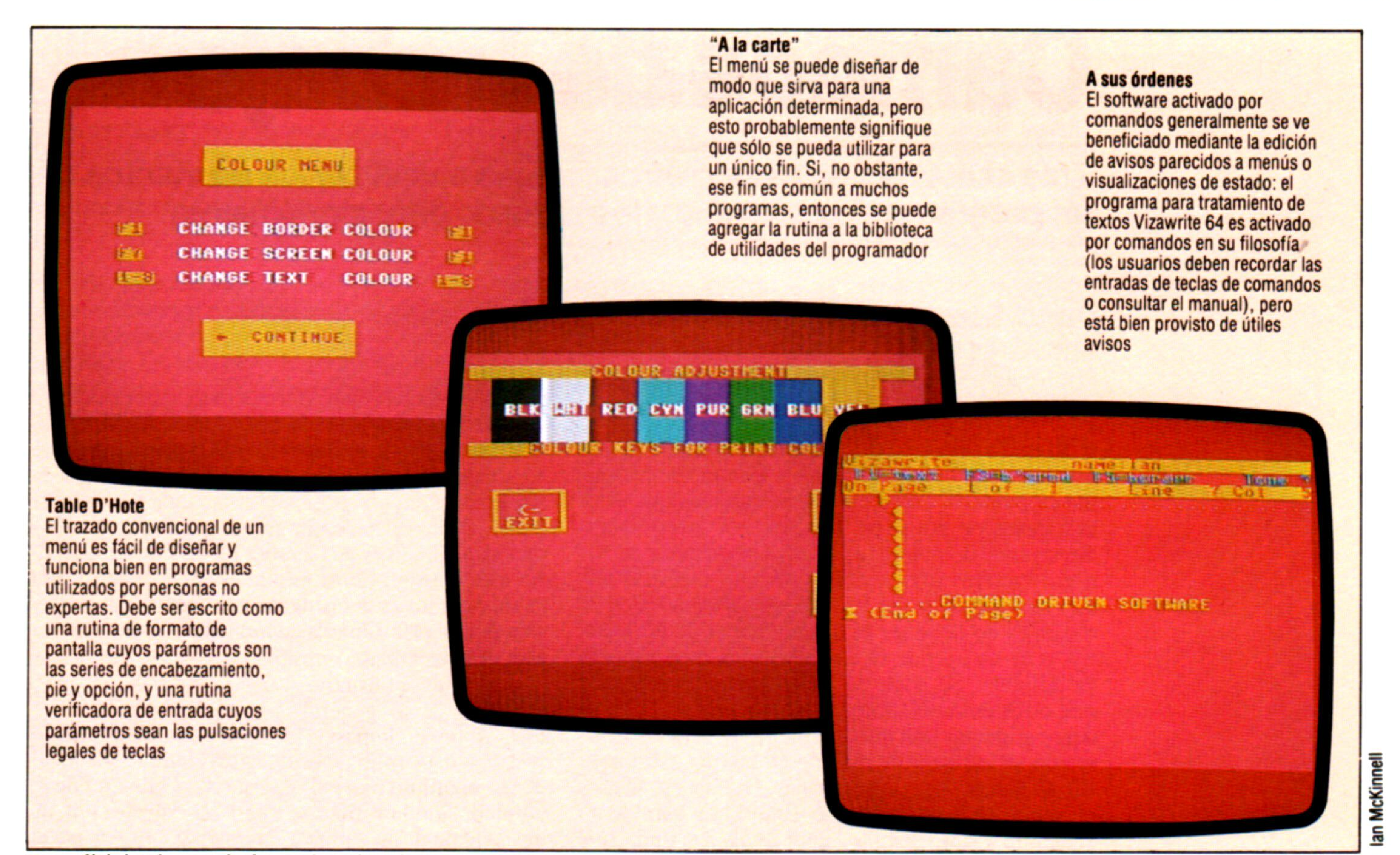
El flujo del programa no seguirá un patrón establecido. Para cada menú a lo largo de la ruta, la rutina de lógica de los menús pasa un conjunto de indicaciones para el usuario a una rutina que las coloca en sus lugares concretos dentro de una "ventana" de menú. Las "ventanas" probablemente contendrán un mensaje de encabezamiento de pantalla que visualice un título y cualquier otra información necesaria acerca del menú, un "pie" que explique cómo hacer la elección (con espacio suficiente para la respuesta del usuario) y las opciones del menú propiamente dichas. En general, el trazado de menú más eficaz posee hasta ocho opciones visualizadas en una columna, con el código de respuesta (número, letra, símbolo mnemónico, etc.) a la izquierda de cada ítem.

La rutina de menús llama a una rutina de entrada, tal vez pasándole las condiciones que se deben especificar para una entrada legal, y acepta a su vez la respuesta del usuario. Luego interpreta esta respuesta (por lo general, una única pulsación de tecla) y o bien le pasa el control al siguiente menú, o llama a la rutina de aplicación apropiada si se trata del último menú de la cadena. Una vez que se ha ejecutado la rutina, se puede visualizar el menú desde el cual fue llamada, o el control puede pasar a alguna otra parte del programa.

Los menús exigen muchísimo texto para encabezamientos, pies e indicaciones, pero gran parte del mismo se repite para cada "ventana" de menú. La explicación sobre cómo elegir una opción del menú (la instrucción de "ayuda": *help*), una opción que ofrezca una salida al menú raíz, y otras opciones posibles puede que se necesiten en varios menús diferentes. De ser éste el caso, se puede ahorrar espacio y hacer que la lógica sea más clara si todas las solicitudes se retienen en una matriz de seres (o en un archivo en disco de acceso directo), desde la cual podrían ser llamadas mediante su número de índice. Diseñe la rutina de visualización de menús de modo que acepte referencias a esta matriz y que visualice los encabezamientos, pies, solicitudes, etc., apropiados.

Un sistema activado por instrucciones es aquel que posee una gama de instrucciones que están a disposición del usuario en todas las etapas del programa. Cada instrucción va directamente a una subrutina que lleva a cabo la función requerida. Este sistema se debe diseñar para inspeccionar todos los input y discenir si se trata de datos o de una instrucción del programa. La diferencia por lo general la señala el usuario mediante la pulsación de una tecla determinada antes de cada entrada de instrucciones. La tecla control se usa frecuentemente para este fin. Un procesador de textos, por ejemplo, puede aceptar la palabra *save* (guardar, salvar) como una palabra más del texto, pero también es posible que la interprete como una instrucción de almacenamiento si antes de digitar la palabra se pulsa la tecla Control.

En un sistema activado por instrucciones, el "árbol" es muy bajo y ancho, y se utiliza una única rutina, que actúa a modo de programa de control,



"A la carte"
El menú se puede diseñar de modo que sirva para una aplicación determinada, pero esto probablemente signifique que sólo se pueda utilizar para un único fin. Si, no obstante, ese fin es común a muchos programas, entonces se puede agregar la rutina a la biblioteca de utilidades del programador

A sus órdenes
El software activado por comandos generalmente se ve beneficiado mediante la edición de avisos parecidos a menús o visualizaciones de estado: el programa para tratamiento de textos Vizawrite 64 es activado por comandos en su filosofía (los usuarios deben recordar las entradas de teclas de comandos o consultar el manual), pero está bien provisto de útiles avisos

Table D'Hote
El trazado convencional de un menú es fácil de diseñar y funciona bien en programas utilizados por personas no expertas. Debe ser escrito como una rutina de formato de pantalla cuyos parámetros son las series de encabezamiento, pie y opción, y una rutina verificadora de entrada cuyos parámetros sean las pulsaciones legales de teclas

Ian McKinnell

Dos en tres
Por lo general existen muchas maneras de resolver el mismo problema; estas visualizaciones muestran distintas formas de permitirle al usuario modificar los colores de pantalla, borde y texto en un Commodore 64

para dirigir al usuario hasta la subrutina requerida. Este "intérprete de instrucciones" tiene cuatro tareas fundamentales. La primera consiste simplemente en esperar una entrada. La segunda, en "analizar gramaticalmente" esta entrada: el intérprete debe separar la línea de entrada en sus unidades funcionales. La tercera tarea consiste en interpretar la instrucción preparando la llamada a la subrutina apropiada. (¿Cuál es la dirección de la rutina? ¿Hay algún parámetro que pasar?) Por último, debe realmente llamar a la subrutina a ejecutar. Cuando el control regresa, el intérprete vuelve a retomar su primera tarea: ¡esperar!

El formato de una instrucción puede ser sumamente elaborado, y algunos lenguajes de instrucciones son similares a una forma simplificada de inglés. Un ejemplo de lenguaje de instrucciones es el esqueleto Unix, en el cual el formato típico de una instrucción es:

Instrucción+lista de parámetros opcionales
Ej.: L
o L−1

Aquí la instrucción Unix L lista un directorio de archivos, mientras que L−1 (donde −1 es un parámetro opcional) lista un directorio de archivos en formato "completo".

El analizador gramatical debe ser capaz de reconocer las diversas partes de la línea de instrucciones. El Unix mantiene las cosas sencillas (en la mayoría de los casos) tomando la primera palabra como la instrucción y reconociendo parámetros mediante un signo "menos" precedente. Los parámetros del lenguaje de instrucciones no son para que los utilice el propio intérprete de instrucciones, sino que los necesitan las subrutinas a las que llama el intérprete. Las rutinas empleadas en el sistema de instrucciones idealmente deberían adoptar un formato estándar para los parámetros de entrada. Si se hace esto, el intérprete de instrucciones puede pasar los parámetros en la forma en la cual fueron entrados (como series, tal vez).

Obviamente, es mucho más sencillo crear un intérprete de instrucciones que escribir un sistema de menús. Los usuarios experimentados tienden a preferir los sistemas de instrucciones o comandos, porque éstos son más rápidos y más flexibles que los programas activados por menú. La mayoría de los sistemas operativos son activados por comandos, lo que resulta lamentable para los usuarios novatos, dado que tales sistemas no proporcionan facilidades de señalización y las rutinas de ayuda en línea (en caso de que haya alguna) exigen cierto conocimiento del sistema. Además, la gran cantidad de instrucciones y de parámetros opcionales de un sistema de comandos típicos significa que incluso quienes estén razonablemente familiarizados con el sistema necesitarán facilidades de ayuda o bien consultar frecuentemente el manual de operatoria.

Los principiantes detestan las instrucciones y los expertos abominan los menús. Este problema es virtualmente insoluble, si bien existen algunos sistemas híbridos que pueden ser bastante eficaces. Por ejemplo, el programa para tratamiento de textos Wordstar es básicamente un sistema activado por comandos, pero al usuario puede parecerle un sistema de menús. Las instrucciones son códigos de control (algunos con parámetros) y el usuario ejecuta el sistema íntegramente con los mismos. Los menús que aparecen en la pantalla utilizan estas instrucciones como mnemónicas para seleccionar opciones de modo que, a medida que el novato va empleando los menús para ejecutar el programa, va aprendiendo al mismo tiempo las instrucciones.

Campos de trabajo

Tras el estudio de los registros, vamos a analizar los elementos de un programa en assembly y los tipos de direccionamiento

Una sentencia en assembly según la encuentra el ensamblador consta de tres partes, aunque no siempre estarán presentes todas. Estas tres partes, o *campos*, son:

- **El campo de etiquetas**, que ocupa la columna que queda más a la izquierda en el listado del programa en pantalla. Las etiquetas no son más que identificadores de números, relativos con frecuencia a direcciones de memoria. Consta la etiqueta de varios caracteres alfanuméricos (de uno a seis): el primer carácter debe ser alfabético, y el nombre en su conjunto no debe ser el mismo que el de un registro, o de un opcode en assembly o de otra etiqueta ya definida. Se trata de puros convencionalismos que otros ensambladores pueden cambiar. Si no se desea etiquetar una línea, al menos debe comenzarse con un carácter de espacio en blanco, para indicar un campo de etiqueta vacío. Del mismo modo, el campo de la etiqueta se concluye con un espacio en blanco; por ejemplo, LABLDA es sin duda una etiqueta correcta, mientras que LAB LDA se deberá interpretar como una etiqueta LAB seguida del opcode LDA.

El ensamblador se apoya en un *contador de posiciones*, que es un equivalente del registro contador de programa utilizado por el procesador. Retiene la dirección de la posición de memoria donde se almacenará el siguiente byte de la instrucción o del dato. En cuanto el ensamblador encuentra una etiqueta, guarda el identificador en un área de memoria llamada la *tabla de símbolos*, muy parecida a un vector en BASIC. Junto con el identificador almacena la dirección que indicaba el contador de posiciones en el momento de descubrir la etiqueta. Inmediatamente después de encontrar una etiqueta en assembly, el ensamblador coteja dicha etiqueta en la tabla de símbolos. Si está en la tabla, el ensamblador se limita a sustituirla por la dirección proporcionada, y si no la encuentra la almacena en la tabla junto con el contenido del contador de posiciones.

- **El campo de operador, instrucción u opcode**, que se sitúa a la derecha del campo de etiquetas. Es una expresión mnemotécnica de tres caracteres por lo general y un nombre de registro en casos necesarios. Por ejemplo, ADDA está compuesta de la expresión mnemotécnica ADD y el registro A. El opcode significa la operación a ejecutar por el procesador. Este campo acaba en un espacio en blanco, como el campo de etiquetas.

- **El campo de operandos o direcciones**, que proporciona información acerca de los datos sobre los cuales operará el opcode. Es frecuente que tales datos se den en forma de dirección, y todavía más frecuente que esta dirección se represente por una etiqueta.

Sea la sentencia en assembly:

```
LABEL1    ADDA    NUM1
```

que quiere decir: "introduce el dato almacenado en la dirección representada por el símbolo NUM1 en el acumulador A, sumándolo (ADD) con lo que éste contenga". La dirección de esta instrucción queda almacenada en una posición etiquetada por LABEL1 (*label*: etiqueta) si deseáramos dar un salto o bifurcación para volver a ella, sólo tendríamos que indicar este destino. NUM1 es una etiqueta especificada ya en otro lugar del programa, y significa la dirección donde está almacenado el dato. Veamos ahora este otro ejemplo:

```
CLRA
```

que significa: "limpia (CLeaR) el acumulador A" (o sea, ponlo a cero). Se trata de un ejemplo de opcode sin operando. Obsérvese que esta línea no lleva etiqueta. Esto se debe a que las etiquetas son de uso opcional, se utilizan cuando se bifurca hacia ellas por medio de cualquier otra instrucción, o bien como una sentencia REM donde se etiquetan líneas significativas con anotaciones (REMarks) explicativas.

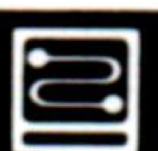
El empleo de etiquetas así resulta de gran ayuda, pero no pueden sustituir a los comentarios más amplios. Estas explicaciones se pueden añadir a cualquier línea dejando un espacio en blanco después del último carácter del operando, y agregando allí el comentario. Algunos ensambladores exigen que haya un carácter especial que les indique el inicio del comentario, y suele ser común comenzar una línea nueva de comentario con un asterisco.

El campo para operandos admite constantes que sean números o cadenas (variables tipo serie). Se suelen dejar los números en su forma decimal, a no ser que se indique por medio del prefijo $ o del sufijo H que están en hexadecimal (así, $AF08, AF08H); o bien se indique por el prefijo @ o el sufijo Q que están en octal (así, @6712,6712Q); o bien, en binario con el prefijo % o el sufijo B (así, %11010011, 11010011B). El código ASCII de un carácter puede servir de operando prefijándole un apóstrofo. Así, 'A significa 65 o $41.

Un valor particularmente útil es el contenido momentáneo del contador de posiciones. Éste no suele conocerse al introducir el programa, pero puede ser referenciado en el campo de operandos con un asterisco. Muchos ensambladores lo admiten en las expresiones aritméticas simples, por lo general la suma y la resta. Por ejemplo:

```
LDA    *+5
```

quiere decir: "carga en el acumulador A el contenido de la posición de memoria cuya dirección es cinco bytes más alta que el contenido actual del contador de posiciones".

Directivas para el ensamblador

El ensamblador suele aceptar diversas *directivas* o *pseudo-ops*, escritas en el programa como opcodes cualesquiera. Ya hemos empleado dos de ellos anteriormente (véase p. 1019):

```
CR        FCB    13
```

FCB (F*ix* C*onstant* B*yte*: fija un byte con valor constante) reserva un solo byte en la dirección indicada por el contador de posiciones, dándole el valor del operando. En este caso, la posición cuya dirección se representa con el símbolo CR queda inicializada con el número 13.

```
MEMTOP    FDB    $7FFF
```

FDB (F*ix* D*ouble* B*yte*: fija dos bytes) hace lo mismo, pero en dos bytes (16 bits). El espacio de memoria se puede reservar por bloques sin tener que fijar el valor de su contenido, con el empleo del pseudo-op RMB. Por ejemplo:

```
TABLE1    RMB    7
TABLE2    FCB    $F6
```

reserva siete bytes para una tabla de valores cuyo primer byte tiene la dirección que indica la etiqueta TABLE1. En la práctica, esto significa que si TABLE1 representa la dirección $C104, pongamos por caso, entonces TABLE2 representará una dirección que es siete bytes más alta, o sea, $C10B.

Toda una cadena de caracteres puede ser colocada en la memoria, como en este ejemplo:

```
ERRMSG    FCC    'ERROR
```

que inicializa cinco bytes de memoria por medio de los códigos ASCII correspondientes a las letras E,R,R,O,R. Es, pues, mucho más fácil introducir mensajes e indicaciones dirigidos al usuario en los programas en lenguaje assembly.

El pseudo-op ORG (origen) es muy importante. Especifica un valor para el contador de posiciones y sirve, al comienzo de un trozo de programa, para orientar al ensamblador sobre el lugar de la memoria donde comenzará a colocar el código del programa cuando se traduzca a código máquina. Siempre se deberían comenzar los programas con una sentencia ORG, aunque algunos ensambladores suplen su falta si no se hace. Se puede, y a veces se aconseja, especificar más de una sentencia ORG en el programa. Una directa ORG no requiere etiqueta ni operando.

La única sentencia que puede que se desee poner antes de ORG es EQU (EQU*als*: igual a), pues especifica los símbolos de las variables y no afecta al contador de posiciones. Por ejemplo:

```
RESET     EQU    $F100
```

significa que el símbolo RESET queda definido como representación del valor $F100. Luego RESET es como la abreviatura o sigla de un número, mientras que una etiqueta colocada al comienzo de la línea de sentencia representa la dirección donde se almacenará el dato o el código máquina.

Finalmente, la última directiva que debemos considerar es END, colocada al final del código fuente como un punto final para el ensamblador. Igual que ORG, no lleva ni etiqueta ni operando.

Tipos de direccionamiento del 6809

Una medida de la potencia de un lenguaje assembly la constituyen sus tipos de direccionamiento, es decir, el número de maneras de interpretar un operando. El 6809 admite muchos de estos tipos. Las instrucciones que hemos visto en los ejemplos se valieron o del modo *directo* o del *ampliado*, que significa que el valor o la etiqueta que se encuentra en el campo de operandos es la dirección de la posición de memoria que contiene el dato.

Direccionamiento directo significa que hay sólo un byte de dirección especificado para el operando de la instrucción. El procesador lo trata como el byte *lo* de la dirección completa del operando que consta de dos bytes, tomando por byte *hi* el contenido del *registro de página directo*, un registro de ocho bits de la CPU direccionable por el programa. Una ventaja de este modo es su flexibilidad y generalidad; por ejemplo, una subrutina puede ser escrita empleando el direccionamiento directo, no haciendo, por ello, referencia explícita a ninguna zona de memoria en concreto. Basta con especificar el contenido del registro de página directo antes de llamar a la rutina para señalar dónde se encuentran los datos en la memoria.

El direccionamiento ampliado consiste en especificar una dirección de dos bytes como operando de la instrucción. Esta instrucción hará siempre referencia a ese byte particular de la memoria, por lo que resulta una pieza inflexible de la codificación. El ensamblador reconoce ambos modos, el directo y el ampliado, por la naturaleza del operando.

Otro modo muy empleado es el *inmediato*, consistente en introducir el dato real en el mismo campo de operandos. Es un modo reconocible por el prefijo # delante del operando. Por ejemplo:

ORG $1000	la dirección inicial del lenguaje máquina es $1000
NUM1 FDB $FFFF	inicializa la posición NUM 1 con $FFFF
LDD NUM1	carga en el registro D el valor $FFFF contenido en NUM1
LDD #NUM1	carga en el registro D el valor $1000 de NUM1 que es el valor real o inmediato.

Se observará que la etiqueta NUM1 representa la dirección $1000. Es posible que usted piense que la instrucción ORG se colocará en la dirección $1000, y que la instrucción siguiente (junto, claro está, con su etiqueta NUM1) recibirán una dirección superior. Pero ha de recordar que ORG es una instrucción que dirige (directiva) el proceso de ensamblado, pero que no forma parte del mismo. Por tanto no ocupa espacio alguno de memoria, lo que explica que NUM1 tome el valor $1000 ya que éste es el valor del contador de posiciones en el momento de ser NUM1 tratado por primera vez por el ensamblador. Debe también observarse que LDD NUM1 carga el *contenido* de la posición NUM1 (que es $FFFF, especificado por la directiva FDB) en el acumulador D, mientras que LDD #NUM1 carga el *valor* propio de NUM1 (esto es, la dirección $1000 de la etiqueta).

Pequeños compromisos

En estos últimos tiempos Casio está intentando acceder al mercado especializado de ordenadores de mano y de bolsillo

Casio afirma controlar el 50 % del mercado internacional de calculadoras pero, sorprendentemente, la empresa cuenta con apenas 3 300 empleados en todo el mundo. En 1983 el volumen de movimiento total de Casio fue de 29 millones de dólares, una cifra baja para un fabricante de electrónica en grandes volúmenes. Tony Manton, director de ventas de calculadoras para Gran Bretaña, aclara que éste es un enfoque deliberado: "Somos una empresa muy pequeña y bastante conservadora. Las grandes campañas publicitarias no son nuestro estilo".

La empresa fue fundada por los cinco hermanos Kashio después de la segunda guerra mundial. En aquel entonces se llamaba Kashio Seisakujo y comenzó fabricando equipos para oficinas. A principios de los años cincuenta, la empresa desarrolló la 14-A Relay Calculator; ésta fue una de las primeras máquinas de calcular eléctricas y era tan grande como el tablero de un escritorio; pesaba 130 k. Se desarrollaron otras calculadoras y en 1957 la empresa cambió su nombre por el de Casio Computer Company Ltd.

Casio se estableció en Gran Bretaña en 1974 y se halló con un mercado fácil para calculadoras electrónicas de precio reducido. En 1982 se lanzó el primer ordenador de bolsillo Casio, el FX-720P. Éste se diseñó para el usuario científico y la disposición de su teclado era inusual, con las teclas dispuestas por orden alfabético en vez de en el tradicional formato QWERTY. Las ventas fueron decepcionantes y la máquina se sustituyó por la FX-700P, que estaba equipada con un teclado QWERTY.

A ella siguió el PB-100, un ordenador tamaño de bolsillo dirigido al usuario de gestión. Similar en diseño al FX-702P, incorporaba una visualización en cristal líquido y un teclado numérico. Casi produjo, asimismo, grabadoras de cassette, impresoras-plotter y paquetes de RAM construidos a la medida para ambas máquinas.

Un reciente sustituto del PB-100 es el FX-750P. Este ordenador configura dos ranuras "RAM-card", que alojan pequeños paquetes metálicos (del tamaño aproximado de una caja de fósforos) y pueden retener hasta 4 Kbytes de datos. Cada ficha está equipada con una pila de tres voltios que almacena el contenido de memoria después que se saca la ficha del ordenador, lo que permite, por lo tanto, guardar y cargar programas en la máquina cuando sea necesario. Cada pila proporciona un almacenamiento de un año; cuando es necesario sustituir la pila, los programas se vuelven a cargar desde cinta.

Casio también produce el FP-200, un ordenador de mano que está equipado con 8 Kbytes de RAM. Ésta se puede ampliar a 32 Kbytes. El FP-200 posee una visualización en cristal líquido que ofrece ocho filas de 20 caracteres en modalidad de textos y tiene una resolución de gráficos de 160×64 pixels. La máquina más nueva de la gama Casio es la SL-800. Se trata de una calculadora de precio reducido que tiene aproximadamente el mismo tamaño y peso que una tarjeta de crédito. El delgado diseño es consecuencia de un proceso de fabricación que se denomina *filmación*, en el cual los componentes se imprimen en película laminada en vez de ser soldados a una placa de circuito impreso.

En muchos países europeos y en el Lejano Oriente, Casio distribuye ordenadores de gestión y ordenadores personales estándar MSX. Sin embargo, en Gran Bretaña la empresa se ha concentrado en calculadoras y pequeños ordenadores. Preguntado acerca de las razones por las que Casio no ha intentado penetrar en el mercado británico personal y de gestión, Tony Manton alude a la naturaleza despiadada de este sector de la industria: "Nuestra intención es ir expandiéndonos hacia arriba lentamente. Es necesario que eduquemos a las personas para que sepan que los ordenadores manuales y de bolsillo son algo más que calculadoras."

Además de cualquier posible expansión en este campo, Casio está aplicando la experiencia adquirida en el campo del teclado electrónico para producir una serie de máquinas alrededor de la interface digital MIDI (véase p. 1014). Dichas máquinas habrán de hacer su presentación en el mercado en un futuro muy inmediato.



Centro de Investigación y Desarrollo de Casio, en Tokio

En la copa del árbol
Tadao Kashio, presidente de Casio, es uno de los cinco miembros de la familia Kashio situados en la cúspide de la estructura corporativa de la firma

Editorial Delta, S.A.



10052
9 788485 822836